Autonomous sailing robot for marine research

Viktor Måsala

School of Electrical Engineering

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 10.4.2018

Thesis supervisor:

Prof. Simo Särkkä

Thesis advisor:

MSc. Anna Friebe

Aalto University
School of Electrical Engineering

AALTO UNIVERSITY
SCHOOL OF ELECTRICAL ENGINEERING

ABSTRACT OF THE
MASTER'S THESIS

Author: Viktor Måsala

Title: Autonomous sailing robot for marine research

Date: 10.4.2018 Language: English Number of pages: 5+45

Department of Electrical Engineering and Automation

Professorship: Translational Engineering

Supervisor: Prof. Simo Särkkä

Advisor: MSc. Anna Friebe

Åland Sailing Robots is a project whose goal is to create an environmentally friendly autonomous marine research vessel, using a sailboat. Autonomous sailing robots bring their own set of problems such as navigating with the wind, and avoiding other boats or rocks.

Åland Sailing Robots outfitted a one man sailing boat with actuators, sensors, and control logic for autonomous sailing. As main propulsion for the boat a wingsail was chosen since it is easier to control than a regular sail. The boat was also fitted with a self-steering mechanism that operates using wind power. The CAN bus was chosen as the main communication protocol between different systems in the robot, since it is robust and new nodes can easily be added or removed. The navigation logic is made up of many simple voters working independently towards their own goal, their votes are added together to achieve the desired course.

The boat won the World Robotic Sailing Championships 2017, where the tests were: covering an area as efficiently as possible, keeping as close as possible to a position for 5 minutes, and a race around buoys.

The structure of the thesis is the following. First some other similar projects are presented shortly, and then a literary study of the technology used in the robot and possible alternatives, focusing on propulsion and communication methods. Then the thesis goes through the specific implementations in the boat in detail, considering solutions and problems that arose. Finally some results from test simulations and from the World Robotic Sailing Championship 2017 are presented.

Keywords: robot, sailing, CAN, autonomous, communication, navigation, propulsion, water, marine

AALTO-UNIVERSITETET
HÖGSKOLAN FÖR ELEKTROTEKNIK

SAMMANDRAG AV
DIPLOMARBETET

Författare: Viktor Måsala

Titel: Autonom robotsegelbåt för marin forskning

Datum: 10.4.2018 Språk: Engelska Sidantal: 5+45

Institutionen för elektroteknik och automation

Professur: Translationell teknik

Övervakare: Prof. Simo Särkkä

Handledare: MSc. Anna Friebe

Åland Sailing Robots är ett projekt vars mål är att skapa en miljövänlig autonom marin forskningsplattform av en segelbåt. Autonoma segelbåtar medför egna problem, till exempel navigering med vinden och undvikandet av andra båtar eller stenar. Åland Sailing Robots utrustade en en-persons segelbåt med aktuatorer, sensorer och styrlogik för autonom segling. Till framdrivning valdes ett vingsegel eftersom det är enklare att kontrollera än ett vanligt segel. Båten förseddes också med en självstyrande mekanism som använder vindkraft. Kommunikationen mellan olika system i båten sker huvudsakligen via CAN-buss, eftersom det är ett robust protokoll och nya noder kan enkelt läggas till eller tagas bort. Navigations logiken består av flera enkla system som fristående röstar för sitt eget mål, deras röster räknas sedan ihop för att uppnå önskad kurs.
Båten tog första plats i World Robotic Sailing Championships 2017, där tävlingsmomenten var: att täcka en yta så bra som möjligt, att hålla sig så nära som möjligt till en position i 5 minuter, och en kapplöpning runt bojar.
Arbetets struktur är följande. Först presenteras några andra liknande projekt kortfattat, sen en litterär studie om teknologin i roboten och möjliga alternativ med fokus på framdrivning och kommunikation. Sen beskrivs implementationen i båten mer detaljerat, inklusive lösningar och problem som uppstod. Till sist presenteras resultat från simulationer och World Robotic Sailing Championship 2017.

Nyckelord: robot, segling, CAN, autonom, kommunikation, navigation, framdrivning, vatten, marin

Acknowledgements

I want to thank advisor MSc. Anna Friebe and Åland Sailing Robots for the opportunity to work on an interesting project, and the guidance and critiques provided. I would also like to thank Prof. Simo Särkkä for helping me in writing this thesis.

Otaniemi, 16.3.2018 Viktor Z. Måsala

Contents

1 Introduction

1.1 Åland sailing robots

Figure 1: The sailing robot during a test sailing. The front servo wing has been removed and replaced with weights to balance the wing. The wind vane is not in use and its main airfoil and the servo rudder is not mounted.

Åland sailing robots is a project at Åland University of Applied Sciences, that started in 2013. Initially the aim was to compete in the Microtransat Challenge [1] and cross the Atlantic with an autonomous sailboat, but it has evolved into creating an environmentally friendly autonomous sailing robot for surveillance and marine research.

The main goal of the project is to increase the competence in unmanned and green technology on Åland islands, and the project is open for, and attracts many international students. The long-term goal is still to complete an unmanned crossing of the Atlantic, and the project currently has funding by European Regional Development Fund for an autonomous marine research platform [2].

Åland sailing robots compete in the World Robotic Sailing Championships (WRSC) [3], and take part in the International Robotic Sailing Conferences (IRSC). The World Robotic Sailing Championship is a competition split into many smaller parts, focusing on a single part of autonomous sailing robots such as navigation, staying still on the water, collision avoidance, and covering as large area as possible in certain time [4]. At the same time when the competition is held the International

Robotic Sailing Conference is also held, a scientific conference where papers are presented on different topics relating to autonomous sailing.

I went to Åland during the summer of 2017 and started developing the CAN bus hardware and software. I also designed the actuator control unit and marine sensor box, and wrote some controller software for the marine sensor box.

1.2 Marine research

At the moment the sailing robot is planned to be equipped with temperature, conductivity and pH sensors for water, as well as sensors for wind speed, direction, air temperature, air pressure, and a Global Positioning System (GPS). This gives a moderate testing suite to monitor weather and water conditions as a proof-of-concept. A hydrophone for detecting harbor porpoises is also planned. The project is in close contact with Husö Biological Station and Åbo Akademi University for feedback about the requirements of the marine sensors and data gathering, and Cornell University for expertise in animal bioacoustics.

1.3 What is this document about

This Master thesis aims to present the sailing robot project as a whole and compare some possibilities and explain the solutions chosen. First some background on the sailing robot and some similar projects in progress are presented. In Sections 2.4 and 2.5 some possibilities for propulsion and communications between systems in the robot are presented and analyzed. These sections are based on research papers and technical documents.

In Section 3, the choices made and how the different components of the boat were implemented are explained in detail. In Section 4, some results from competitions, simulations, and test sailings are presented.

2 Background

2.1 Why an autonomous sailboat

Since the robot is meant to be out at sea gathering data for long periods of time, it should use as little energy as possible. A sailing robot platform was chosen because of its low energy consumption and therefore it has the potential for cost effectiveness and environmental friendliness. If it is traveling in a straight line, and if the wind does not change, it uses no other energy for the propulsion than the sailing wind. The boat will use solar power for other energy needs, and wind and water power generation could potentially also be used.

For a research vessel of this type speed is not really an important factor, only gathering as much data as possible over long periods of time. For marine research the important things are long-term deployment and robustness. It is also important for the vessel to be able to handle shallow waters since large boats and underwater vessels cannot go there. This was the response of a small survey that had been sent out to potential users of this kind of technology around the world such as to researchers, people who make nautical charts, surveillance companies, and others who might be interested. According to the survey answers speed was a low priority, while endurance and ability to function in adverse weather was ranked highly.

The boat can also be used as a stationary measuring station without the need for anchoring it to the sea floor. While deep ocean buoys can drift far away from their anchored position since their anchoring lines are so long, a wind and solar powered vessel could keep its position for very long periods of time [5]. Another potential for a boat is the ability to carry larger payloads for lower cost than an unmanned aerial vehicle, and longer operating times since planes need to land occasionally.

2.2 Autonomy

Being autonomous brings a lot of problems to solve, but is also comes with benefits. Since it is unmanned, there is little or no cost for crew, and it can operate at all hours of the day. The cost of renting a research vessel with crew can be between 3400 € and 26000 € per day [6][7][8][9]. The sailing robot is planned to be able to follow predefined waypoints gathering environmental data, and doing adjustments to its course to avoid collisions based on data from the Automatic Identification System (AIS) and thermal camera. They could be things like the rocks, people, and boats. The sailing robot also needs to use different sailing techniques, such as tacking and adjusting the wingsail, based on wind speeds and directions relative to its course, to be able to navigate properly at sea.

2.3 Other similar projects

The idea of an autonomous sailing robot for marine research is quite new, but there are already many companies developing their own versions. In the following we list some of the more prominent ones.

2.3.1 Saildrone

Saildrone [10] is an advanced wingsail powered trimaran designed for ocean research. It began with founder Richard Jenkins trying to, and succeeding in breaking the wind powered land speed record in 2009. In doing so, he designed an efficient wingsail. He later realized that this wingsail would be suitable for autonomous sailing robots, since it is much easier to control than a regular sail, as it effectively has only one control for setting the speed of the vessel.

Figure 2: The Saildrone. The image is from [11].

Saildrone is equipped with 16 different sensors that can measure wind speed, wind direction, sunlight, air temperature, pressure, and humidity. For aquatic measurements it can measure magnetic fields, temperature, ocean currents, dissolved oxygen, salinity, and marine animal acoustics among other things. It is especially interesting due to the fact that it has already completed many research missions where it has sailed autonomously for months before returning to the shore. Most autonomous sailing robots are still quite early in the development stage.

The wingsail on Saildrone has one servo wing at the back controlling the angle of the sail, and a counterweight in front balancing it and allowing it to rotate freely. The speed of the vessel is on average 3-5 knots.

2.3.2 Atlantis project

The Atlantis project [5] is a well-documented early autonomous wingsailed catamaran project from Stanford University which was started in 1997. In the thesis by Gabriel Hugh Elkaim he goes into great detail in the design and construction of the boat.

The thesis contains a lot of equations and operating principles for an attitude system for determining precisely the current orientation of the boat, the design and operation of wingsails, and on how to model the boat for control purposes. The thesis also contains experimental data showing how precisely the position of the boat could be measured, and how well the boat could be controlled. The university managed to get a precision of 0.3 meters standard deviation when following a line.

The boat had a CAN bus connecting all the different motors and sensors of it, and the main controller was a regular laptop.

2.3.3 C-Enduro

The C-Enduro [12] is a catamaran that is powered by 2 electric motors with propellers. It has a wind turbine, a diesel generator, and solar panels for energy generation. It is designed for data collection with variable payload. According to the website it can handle missions that are over 30 day long. It can be configured with waypoints and can autonomously navigate between them, or can be controlled directly.

Figure 3: The C-Enduro. The image is from [12].

2.3.4 AutoNaut

The AutoNaut [13] is an autonomous vessel that is wave-propelled. The forward motion comes from fins in the bow and stern of the boat. When waves moves the boat up and down, water causes the fins to tilt, creating an angle relative to the water and giving them an angle of attack producing forward thrust [14] (see Figure 4). The electronics on the boat is powered by solar panels that charge batteries.

The AutoNaut is produced in four sizes; 2, 3.5, 5 and 7 meters in length. The 3.5 m AutoNauts speed is claimed to be 3 knots, and according to their site the speed scales with the size, so the larger versions move faster. The boat can navigate by itself between waypoints, using an AIS for collision avoidance. The boat can be communicated with and waypoints changed via satellite or Ultra High Frequency (UHF) signals, or the boat can be controlled directly.

2.3.5 Wave Glider

Wave Glider [15] is also propelled by wave power in a similar way to the AutoNaut, but instead of the fins being attached to the boat itself, they are placed on a separate "sub" that is attached to the boat with a cable. When the sub moves up and down, it is propelled forward pulling the boat with it. The sub is also equipped with an

electric motor for extra propulsion when needed. The speed of the vessel is up to 3 knots.

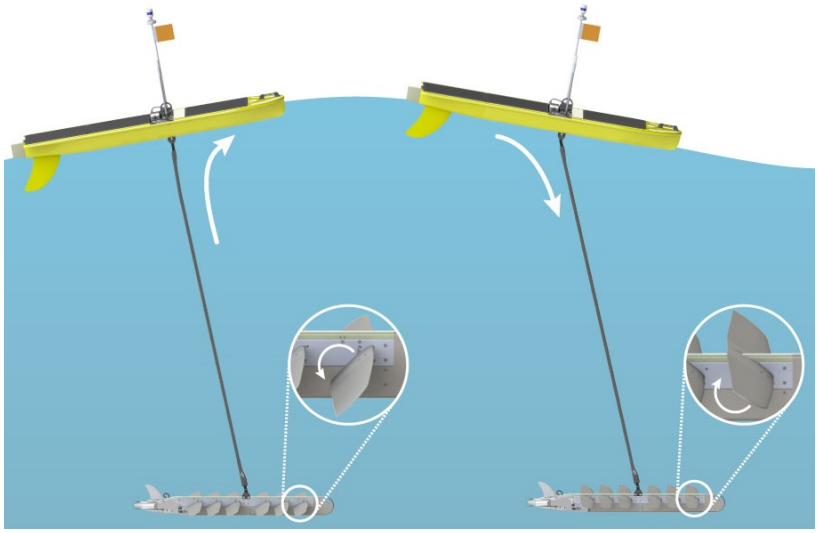


Figure 4: Wave propulsion operating principle, here showing the Wave Glider. The image is from [16].

One downside of this configuration is that the boat cannot go into shallow waters (<8 m) since the sub would hit the seafloor. Like most other autonomous boats it is equipped with solar panels for powering the navigation computer and sensors.

2.3.6 Submaran

Ocean Aero is developing an autonomous sea vessel Submaran [17] that can go both on the surface and underwater. It is propelled by a wingsail at the surface and an auxiliary thruster. The wingsail can be folded down when submerging. It has solar panels for recharging it batteries, and powering the thrusters and sensors. It can manage months at sea.

2.4 Main propulsion methods

In this section some common methods of propulsion for boats is listed, to give an overview of some of the possibilities for an autonomous robot boat.

2.4.1 Sheet sail

Sails have been used for a long time to power boats at sea. Paintings, models, and remains of boats with masts have been found in Mesopotamia dating from the Ubaid period (6000-4300 BC) [18].

Modern sails operate by having a sheet of fabric stretched out and attached to a mast in the boat. It works by the same principles as an airplane wing. When air

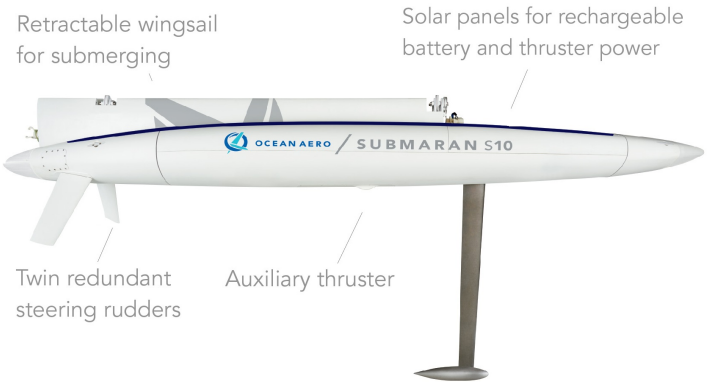


Figure 5: The Submaran. The image is from [17].

is blowing on the sail the fabric creates an airfoil shape. Air moving past the sail moves faster on the backside of the sail, creating a pressure difference resulting in a lift force on the sail (see Figure 6). This is called the Bernoulli effect [19]. The lift can also be explained using Newton's laws, when air is redirected by the sail it creates an equal and opposite lift force on the sail [20].

The sail also causes drag in the same direction as the wind. When combined with the lift the total force is pointing somewhat forward and away from the wind. The keel of the boat has to counteract the component that is not pointed forward. [19]

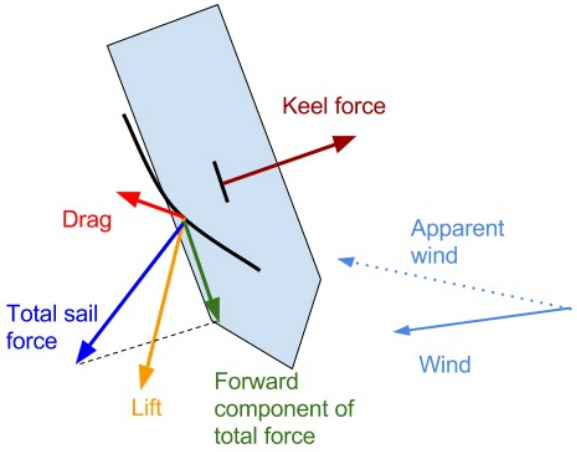


Figure 6: Sail forces from wind.

The wind that affects the sail is the wind vector minus the forward vector of the boat, this is called the apparent wind. This makes a wind coming straight from the rear worse for sailing than a wind straight from the side [19]. With the wind from the rear, the boat can never move faster than the wind speed, while if the wind is coming from the side the apparent wind speed is almost constant regardless of the speed of the boat. It is possible to sail somewhat against the wind, but if the angle becomes too small, the sail, since it is a fabric, will lose its rigidity and start flapping

in the wind, called luffing, losing its wing-like properties.

When sailing upwind a technique called tacking [21] is used. When tacking instead of moving straight into the wind, the boat moves at about a 40 degree angle to the wind alternating at which side it keeps the wind on. This way the boat does not go too far off course.

When sailing, the angle of the sail must be adjusted according to the direction of the apparent wind constantly to keep the speed up. This is usually done using winches and ropes attached to the beam of the mast. When using electric motors to do it they have to be quite fast and strong.

2.4.2 Wingsail

Instead of using fabric as a sail, a rigid wing can be used. Wingsails work on the same principle as an airplane wing [22] and a regular sail, except it keeps its shape in all kinds of winds.

It can either be controlled directly or be rotating freely controlled by a servo wing, a smaller wing attached to the main wing some distance away from it. When the servo wing is rotated, the wind makes the whole wingsail assembly turn slightly, causing the main wing to generate lift. When controlled directly it has the same problem as a regular sail for a robotic boat: they are complicated to keep in the correct position in changing wind. With a free-rotating wingsail, the problem of keeping it in the correct position disappears. One is simply left with a single variable to control, the angle of the servo wing which acts as a simple throttle control [23]. This makes a wingsail much simpler to control since the wind direction need not be accounted for, except for making sure that the boat is not trying to sail directly against the wind. Sudden changes in the wind will also be handled automatically.

To make a free-rotating wingsail behave correctly the axis which it rotates around should be in the aerodynamic center of the wing, which is one quarter of the chord (line from leading to trailing edge of wing) from the leading edge [22]. The center of gravity should also be in the same spot to minimize the force required to turn the wing, so it functions correctly even with low wind speed.

Due to its aerodynamic shape a wingsail aligned with the wind also has lower drag than a bare mast [5] of a regular sail. This also makes it possible to not have to take down the sail in strong winds.

There are many different configurations of wingsails, servo wings, and flaps. The most common design is a servo wing behind the wingsail. The servo wing can also be placed in front of the wing. It can also be free-floating and have its own small flap controlling it. The main wing may also have a flap in addition to the servo wing, for additional control of the angle. [5]

The main benefit of having the servo wing in front of the main wing is that the radius from the mast is smaller than if the servo wing is behind the main wing, since the mast will be attached towards the leading edge of the wing. The center of mass will also be closer to the aerodynamic center of the wing. [5]

2.4.3 Powered sails

Flettner or Magnus rotors are powered vertical rotors that when spinning creates a lower pressure on one side than the other, creating lift [24] (see Figure 7).

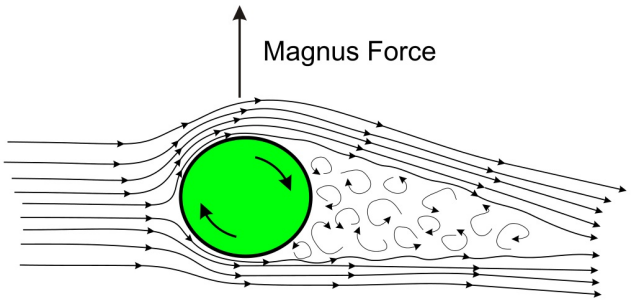


Figure 7: Magnus effect; air moving around the rotor will have a higher speed on the bottom side than the top. This results in lower pressure on the top side than the bottom, creating the upward force. The image is from [25].

Compared to non-powered sails the Flettner rotors have a high lift coefficient relative to their size [24][26]. They are also structurally stronger than wing- or sheet sails, since a cylinder is a strong shape. They are also smaller and weighs less than sails or wingsails with comparable lift. [24]

Since Flettner rotors are completely symmetrical, they have no direction to adjust, and always produce lift perpendicular to the wind direction, and drag in the same direction as the wind. This means sailing directly upwind is not possible, and notably not directly downwind either except for a relatively small drag coefficient. When tacking the direction of rotation needs to be reversed. By adjusting the speed, the maximum lift can be limited, which means that even if the wind speed increases dramatically the lift remains virtually unchanged. [24]

Still since Flettner rotors need external power to function, all in all they were not as efficient as wingsails when tested [24] as assistance to regular propeller power as a fuel saving measure. They were about as efficient as regular sails, when factoring in the fuel cost of powering the rotor.

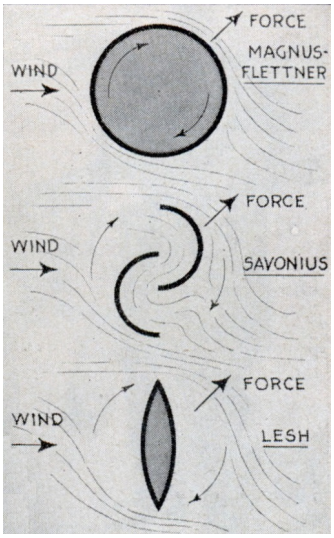


Figure 8: Comparison of different Magnus effect wind rotors. [27]

There is also a possibility to create a Flettner rotor that powers its own rotation [28], but it requires an external force to start. There also exists auto rotors such as the Savonius and Lesh rotors, wind turbines that generate the Magnus effect when spinning [28][29][30], but not

much research can be found about them.

The Savonius rotor would have to have some potentially complicated mechanics shifting the halves of the rotor to reverse the direction of spin for tacking maneuvers, while the Lesh rotor works in both directions [30]. But both the Lesh and Savonius rotors lack any control at all, making them not ideal for a boat where precise control in varying wind speeds is needed.

2.4.4 Propeller

A common way to propel a boat forward is to use a propeller. For it to work in an autonomous vessel that is supposed to be out at sea for long periods of time, it needs to be powered by a resource that can be gathered from the boat's environment, such as electricity from solar panels, wind power, or water turbines. Otherwise it would severely limit the operating time.

A propeller could cause noise, affecting marine mammals and disturb acoustic recordings, or unnecessary muddle the water possibly affecting sensor readings. But if there is no wind an electrically driven propeller could be a good way to move if needed for some reason, such as in emergencies to avoid collisions.

Some research [31][32] has been done about wind turbines powering propellers on boats. In [31] Blackford experiments with a wind turbine mechanically coupled with a wheel on a small vehicle and a propeller on a small boat. The land vehicle achieved an upwind speed matching the wind speed, while the boat achieved half the wind speed. He also noted that for wind turbine propulsion to give more force than a wingsail for its size, the wind speed needs to be at least twice the speed of the boat.

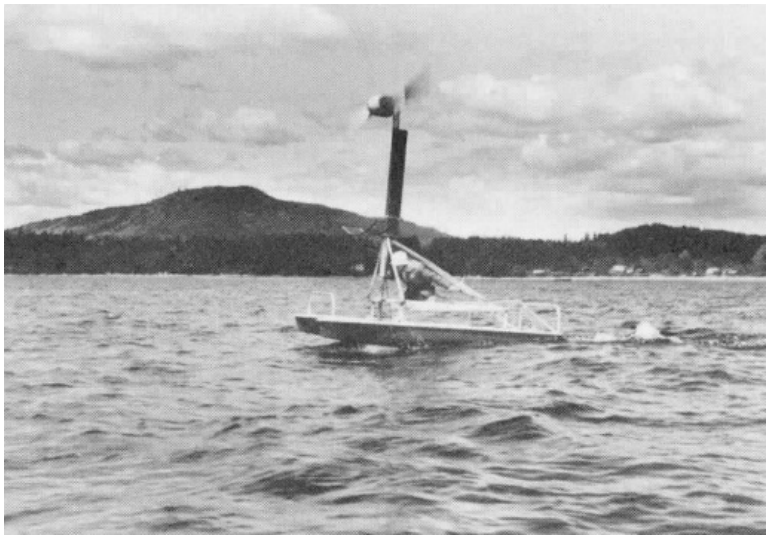


Figure 9: Photograph of Blackfords experimental 4 m windmill boat. The image is from [31].

In [32] Eirik Bøckmann and Sverre Steenthey discuss the possible fuel saving of tanker ships equipped with wind turbines generating energy for the propulsion

compared to wingsails. Their conclusion is that the turbines and wingsails had comparable performances in their test case.

2.5 Communication between systems

This section will cover some commonly used electronic communication protocols. It will go into detail how they work and compare them to each other, particularly in applications to the sailing robot.

2.5.1 SPI

Serial Peripheral Interface (SPI) [33] is a full duplex serial communication system, meaning that it can both receive and transmit at the same time. There is no formal specification for SPI so it has no set data rate or communication protocol. It is up to individual manufacturers to define their own parameters and commands. SPI was developed by Motorola and this section is based on their guide [33].

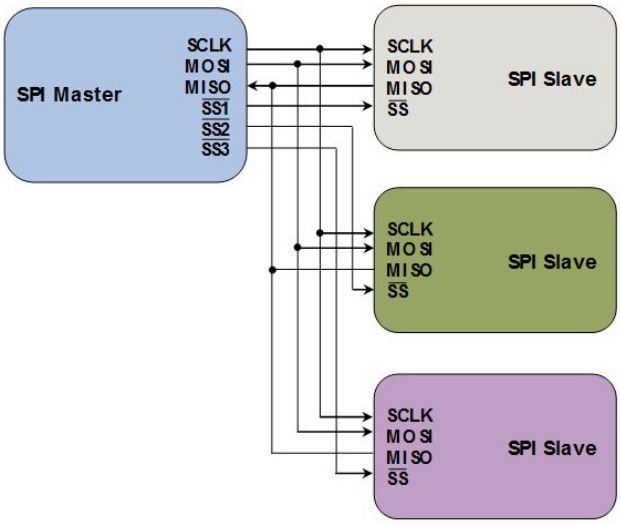


Figure 10: SPI diagram. The image is from [34].

SPI has 3 wires for transmitting data. Master Out Slave In (MOSI) line for data sent from the master to the slave, Master In Slave Out (MISO) line for data sent from the slave to the master, and Serial Clock (SCK) line for a clock signal controlled by the master telling the slave device when to read/write data. There is also a Slave Select (SS) line for each device that the master wants to communicate with, which is used to select which device to communicate with at any moment. When a clock signal is sent, data is both transmitted and received at the same time through MOSI and MISO, to and from the selected device.

The system can have only one master active at a time, and the number of slaves is limited by the number of SS pins on the master.

Table 1: The four SPI transfer formats

CPOL	CPHA
0	0
0	1
1	0
1	1

Even though there is no set specification when data should be sampled, the four transfer formats defined by clock phase (CPHA) and clock polarity (CPOL) that typically are used can be seen in Table 1. These transfer formats define the sampling point according to Figure 11. When CPHA is 1, data is sampled on even edges, in other words the edge following the clock pulse. With CPHA set to 0, sampling occurs on odd edges, or the first edge of the clock pulse. CPOL decides the idle and active states of the clock signal. CPOL 0 means that the clock is low when idle and the clock pulse is high, and CPOL 1 is idle high and clock pulse low. [33]

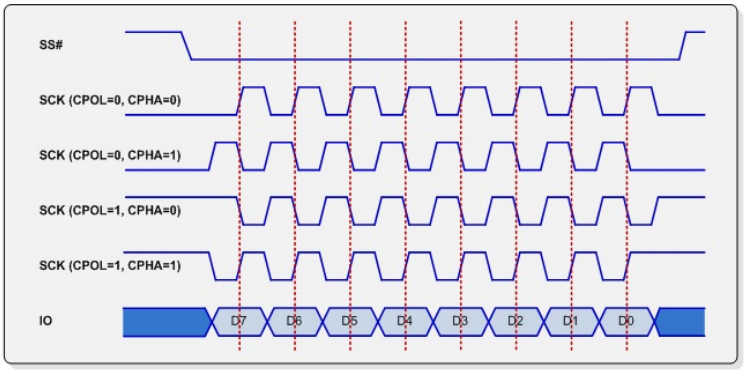


Figure 11: SPI sampling point according to transfer formats. The image is from [35].

Since SPI has no message protocol, it is a very flexible communication system, any size messages can be sent. It is also very easy to implement on a microcontroller if needed.

2.5.2 I^2C

Inter-Integrated Circuit (I^2C) [36] is a two-wire bus invented by Philips Semiconductor for communication in electronics, and it can support multiple masters. It has similar intended application areas as SPI, in that it is made primarily for direct communication between IC chips. From the specification for the bus [36], it is a two-wire bus that has a Serial Data Line (SDA) and Serial Clock Line (SCL) pulled up by resistors. The nodes can only pull the lines low, this is important for when multiple nodes try to transmit at the same time or for clock stretching.

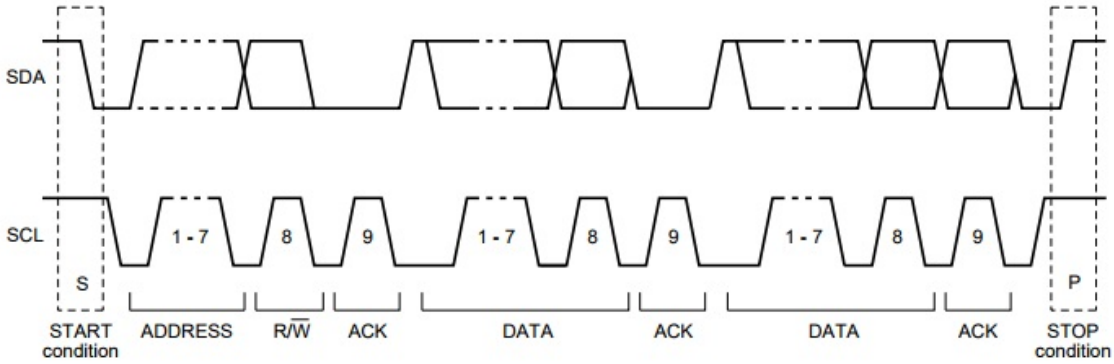


Figure 12: A complete I^{2}C transfer. The image is from [36].

Data transfers, (see Figure 12) on the bus are always controlled by a master. The master generates the clock and initializes the transfer, either sending data or requesting that other nodes send it data. A node wanting to transmit data on the bus sends a START condition on the bus by pulling SDA from high to low while keeping SCL high. If the system has many masters arbitration occurs. Following the START signal comes a 7-bit slave address and a data direction bit (R/$\bar{\text{W}}$), where low indicates write and high indicates read.

Data is transferred one byte at a time, followed by an acknowledge bit (ACK). During the ACK bit the transmitter lets the SDA line float high, and the receiver takes control of it, the transmitter still has control of the SCL line. Pulling the SDA low during this clock signal is the "Acknowledge" signal and letting it be high is the "Not Acknowledge" signal. If the receiver is not ready to receive another byte directly after the previous it can hold the SCL line low to force the master to wait. This is called clock stretching.

When the transfer is completed the node sends a STOP by letting SDA go from low to high while SCL is high, or it can send another START to start another transmission immediately.

If there are multiple masters on the bus clock synchronization and arbitration needs to be implemented. Clock synchronization works by having the master with the longest low period hold the SCL low and others will wait, when the SCL goes high the master with the shortest high period will pull the SCL low when it is done.

Arbitration happens if two or more masters start transmitting at the same time. All masters check if the SDA level matches what they want to send if it is a mismatch the master stops trying to transmit, and wait for the bus to be free again.

The transmission speed of I^{2}C varies with what mode it is operating in. Standard mode is up to 100 kbits/s, fast mode is 400 kbits/s, fast mode plus is 1 Mbits/s and high-speed mode is 3.4 Mbits/s.

The length and maximum number of nodes are limited by the capacitance of the system, recommended are 400 pF per bus line for standard and fast mode, 550 pF for fast mode plus. This is because of the RC constant that the bus capacitance and resistance and pull up resistor creates. This transition has to be within spec for the

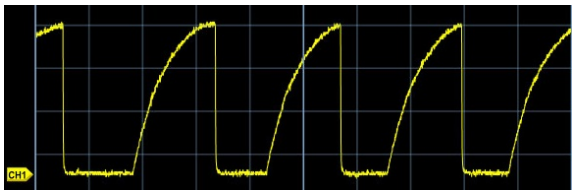


Figure 13: I²C clock signal with a very large pull up resistor, 70K Ω, viewed in an oscilloscope. Shows a clear saw-wave instead of the desired square-wave. The image is from [37].

data to be read properly. A larger RC constant will slow the transition and turn the square-wave into a saw-wave, (see Figure 13). This can lead to failure to detect the correct logic state. [36]

2.5.3 CAN

Controller Area Network Bus (CAN Bus) [38] is a serial communication protocol designed by BOSCH. It uses a two-wire bus and supports multiple masters. It is designed for longer distances than SPI and I²C, and is therefore more robust. The physical cable consists of a twisted pair (CAN High and CAN Low) terminated with 120 Ω resistance in both ends for the differential signal and two wires for powering devices on the bus.

Signals are sent as a differential voltage for improved noise immunity. When sending equal but inverted signals through the wires, and measuring the difference induced noise will cancel out. When sending a dominant bit (logical 0) the transceiver drives the high line towards 3.5 V and the low towards 1.5 V creating a difference of 2 V. On a recessive bit (logical 1) the transceiver floats the outputs and the terminating resistors draw the lines toward an equal voltage level $\sim$ 2.5 V. [38]

If two nodes try to send a message at the same time, a bitwise arbitration, the same way I²C does arbitration, occurs on the identifier of the message. When a node sends a bit, it reads the bus to check if it gets the same value back, since only logical 0 is driven if one node tries to send a 0 and another tries to send a 1 the 0 will drive the lines while the 1 is passive. If the node notices that it reads a different value than it sends it will know that another is trying to send at the same time and will stop sending and wait for the other to be finished. The message with the highest priority (lowest identifier) wins the arbitration and is sent. A node that stopped transmitting will try again when the bus is available [39]. This is part of the ISO 11898 physical signaling layer [38] so it is implemented in most CAN controllers.

There are two types of CAN messages, or frames, as they are called; the standard frame and the extended frame. The difference is that the standard frame has an 11 bit identifier and the extended frame has an 29 bit identifier [39]. The standard 11 bit identifier gives $2^{11} = 2048$ unique message identifiers while the extended 29 bit identifier give $2^{29} = 537$ million identifiers.

There are 4 types of frames that can be sent on a CAN bus. Data frames are

meant for transmitting data on the bus. A Remote frame is for requesting other nodes to send the Data frame with the same identifier as the Remote frame. An Error frame is sent when a node detects an error. And lastly Overload frames are used as delays between frames. [39]

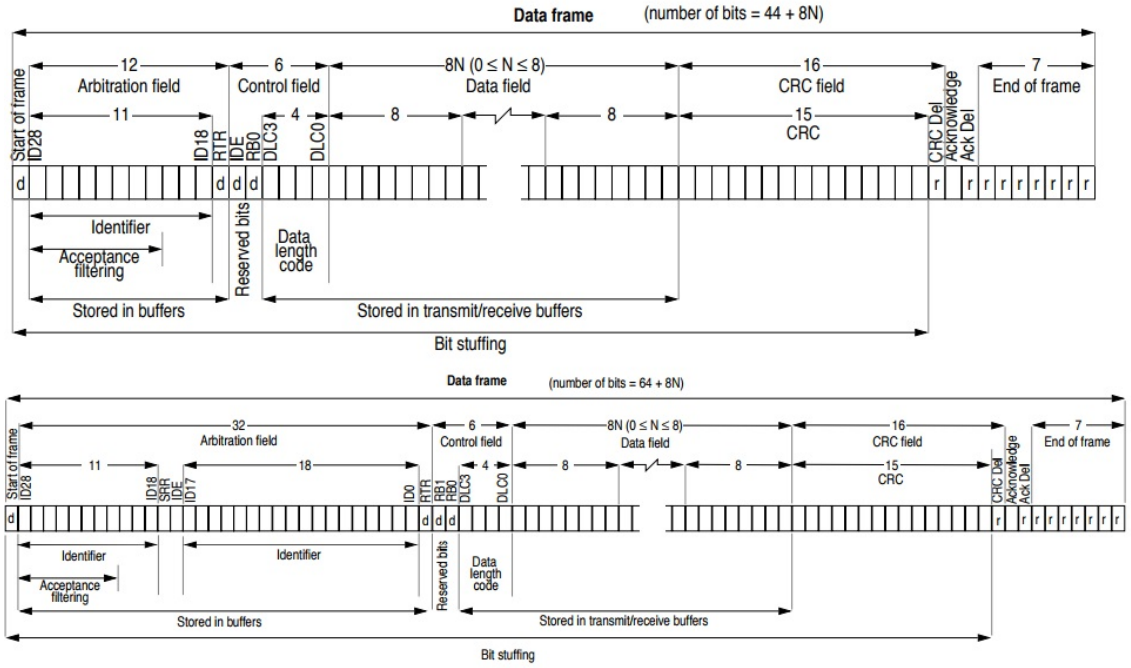


Figure 14: The layout of standard and extended CAN frames. The image is from [40].

On the CAN bus messages are sent most significant bit (MSB) first. The messages begins with an Start Of Frame (SOF) bit which is a single dominant bit. The first proper part of the message is called the Arbitration field.

In a standard frame the Arbitration Field consists of the 11 bit identifier and Remote Transmit Request (RTR) bit. The RTR bit denotes if the message is a remote frame. In an extended frame the Arbitration field contains the 29 bit identifier and a Substitute Remote Request (SRR) bit, a Extended identifier (IDE) bit (recessive) and the RTR bit. In the extended frame the identifier is split up into a base identifier 11-bits and extended identifier 18-bits, the base identifier is the most significant bits of the identifier. (See Figure 14.)

After the Arbitration field comes the Control field, which in standard frames consists of the Data Length Code (DLC), Extended identifier (IDE) bit and a reserved bit which has to be dominant (logical 0). In extended frames the Control field contains the DLC and two dominant reserved bits. The DLC is 4 bits long and tells the receiver how many bytes of data the message has. Although the maximum value of a 4 bit number is 15 the maximum amount of data a message can have is 8 bytes.

Following the Control field comes the Data field. It contains the data in the message. It has a variable length between 0 and 8 bytes, as told by the DLC bits.

After the Data field comes the Cyclic Redundancy Code (CRC) field that contains a CRC sequence and a CRC delimiter. The CRC sequence is a checksum of the number of bits sent thus far for error checking. The CRC delimiter is a recessive bit. The ACK field contains the ACK slot and a ACK delimiter, both are recessive. A receiver that has received a valid message sends a dominant bit during the ACK slot, overwriting it, to signal the transmitter that the message was valid. The message ends with an End Of Frame (EOF) sequence consisting of seven recessive bits, denoting the bus is available again. [38][39]

2.5.4 J1939

J1939 is a common CAN message protocol used in heavy-duty vehicles. An overview can be found in Vectors Introduction to J1939 [41], the bit rate is 250 kbit/s and it uses the extended CAN message identifier and splits the 29 bits as follows: 3 bits for Priority, 1 bit for Reserved, 1 bit for Data page, 8 bits for Protocol Data Unit (PDU) format, 8 bits for PDU specific, and 8 bits for Source address. Priority is used for message arbitration, since the first node to transmit a 1 (recessive bit) will lose arbitration so lower priority values win.

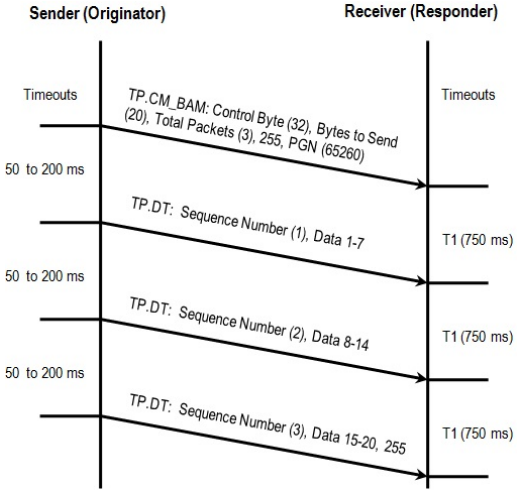
If PDU format is between 0 and 239, the message is addressed to a specific node, and the PDU specific contains the address. Address 255 is used for as a global address when sending messages to all nodes on the bus, and 254 is a NULL address used as source address for error messages. If the PDU format is between 240 and 255 the message is broadcast to all nodes and the PDU specific is used to expand the number of available unique messages. Source address is the address of the node that sent the message.

When dealing with messages using J1939, one commonly refers to them using a Parameter Group Number (PGN). The number is constructed by combining the Reserved bit, Data page, PDU format, and PDU specific into a single 18 bit value [41][42]. Even if it is addressed to a specific node the address is considered 0 for the sake of the PGN.

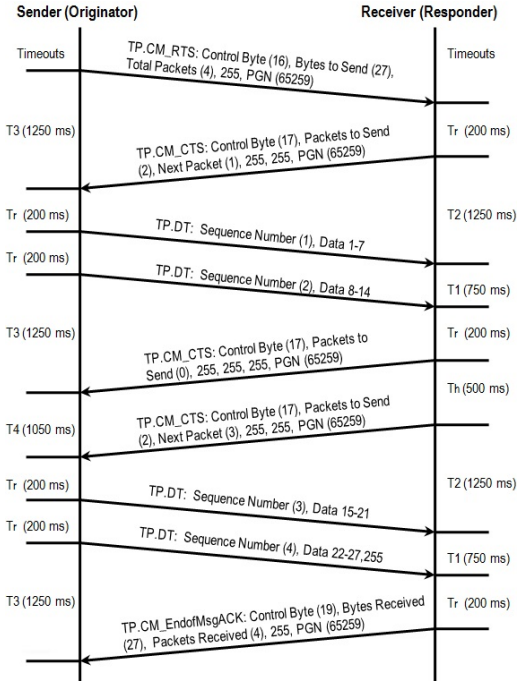
When a J1939 node connects to the bus it has to claim a unique address on the bus. Every node sends an address claim message with its address, and the other nodes responds if the address has already been claimed by a node with higher priority. The name sent as data in the address claim is used as priority.

Another possibility is to send a request for address claim, where all other nodes must respond with their claimed addresses and the new node can choose a free address to claim. The devices can be Arbitrary address capable or Single address capable. If the device is Arbitrary address capable the device itself handles the address change if another node has the address it wanted, if it is Single address capable the device can not change its address by itself.

J1939 has its own protocols for sending messages that are larger than 8 bytes, where the messages must be split up into 8 byte packages. One is Broadcast Announcement Message (BAM), see Figure 15a, for sending to all nodes on the bus. First it sends a control byte with message info including the number of bytes to send and total packages followed by the packages with a small pause between them. The



(a) J1939 protocol CAN Broadcast Announcement Message transfer example, showing the data flow from the sender to the receiver. The image is from [43].



(b) J1939 CAN Connection Mode transfer example, showing the back and forth data transfers. The image is from [43].

other is Connection Mode (CM), see Figure 15b, for sending a large message to a specific node on the bus. Here the transmitter must send a Requests To Send (RTS) and the receiver must reply with a Clear To Send (CTS) before the transmission can begin. When the receiver has received the full message it sends a "end of message acknowledge" message. [41]

2.5.5 NMEA 2000

NMEA 2000 is a message protocol used in marine communications derived from the J1939 protocol, and it uses the same message format and speed (250 kbits/s) [44][45]. The CAN standard sets the maximum theoretical length of communication at 200 meters at 250 kbits/s based on signal propagation delay [45]. But voltage also limits the max length of a cable. Since NMEA 2000 devices are made to function with voltage between 9 and 16 volts [45] the maximum voltage drop allowed is 3 volts if using a 12 V battery. Manufacturers must provide a Load Equivalency Number (LEN) for their NMEA 2000 devices that specifies how much power it draws from the bus, 1 LEN is equal to 50 mA [46].

NMEA 2000 also implements a new message protocol called fast packages [45], allowing 256 bytes of data to be sent in rapid succession. The first byte contains an identifier for the package and a frame counter. The first message also contains in the second data byte the total number of bytes in all of the packages.

2.5.6 USB

Universal Serial Bus (USB) [47] is a standard that defines cables, connectors and communication protocols for sending data to and from a PC. USB supports up to 127 devices on the bus, and a USB port supplies 5 volts and up to 500 mA power. The host controller initiates all data transfers. The host is connected to all other devices through hubs, starting with the root hub on the computer's motherboard that is connected to the USB connectors. Other hubs may be connected to these. [47]

Transmission speed was 12 Mbits/s (full-speed) and 1.5 Mbits/s (low-speed) in the first USB standard, USB 2.0 increased it to 480 Mbits/s (high-speed), and in USB 3.0 it was increased further to 450 MBytes/s and in USB 3.1 the speed is 1 GByte/s. All the standards are backwards compatible. [48]

USB 3.1 is a parallel bus where there is both a USB 2.0 and an Enhanced SuperSpeed interface. If the device supports Enhanced SuperSpeed it will use it and otherwise it falls back on USB 2.0 speeds. The USB 3.1 cable has outside the standard 2.0 D+ and D- twisted signal pair and power two pairs of twisted signal wires for Enhanced SuperSpeed, one for transmit and one for receive. [48]

The main benefit of the USB standard is plug and play, the ability to plug in and unplug devices at any time without any problems. This is not really relevant in robotics, since the hardware configurations tend to be static while the machine is running.

If USB is needed in custom hardware designs the best way is to get a premade chip for it such as one from FTDI [49] or equivalent. There are many chips that emulate a serial (COM) port such as FT232R [50], CH340 [51] and CP2102 [52]. Another possibility is using a microcontroller with builtin USB support.

2.5.7 RS-232

EIA/TIA-232-E [53] is a standard for serial communication commonly known as RS-232. RS stands for Recommended Standard and was developed by the Electronic Industry Association and the Telecommunications Industry Association (EIA/TIA). The official name of the standard is "Interface Between Data Terminal Equipment and Data Circuit–Termination Equipment Employing Serial Binary Data Interchange". The standard defines communication between a host system, known as a Data Terminal Equipment (DTE), and a peripheral system, Data Circuit-Terminating Equipment (DCE). This was the main standard in computers before USB replaced it.

Since the standard was introduced in 1962 it does not use TTL voltage levels (Low: 0-0.8 V High: 2-5 V) as IC chips use today. A high voltage level is +5 V to +15 V and a low level is -5 V to -15 V, but the receivers has a noise margin of 2 V making the high level be from +3 V to 15 V and low from -3 V to -15 V [53][54]. In the standard a high voltage represents a logic 0 and a low voltage represents a logic 1. The standard also specifies a maximum slew rate of 30 V/µs and a maximum data rate of 20 kbits/s, for reducing crosstalk between adjacent signals [53]. The maximum data rate was in later implementations increased [53][54] to 100-300 kbits/s. The

load the transmitter sees is 3 to 7k Ω and the max capacitive load of the cable is 2500pF according to the standard [53][54].

The standard also specifies the signals used in communication, see Table 2. Few devices use all of these. For simple data transfer only 5 wires are needed: Transmitted Data (TD), Received Data (RD), Request to Send (RTS), Clear to Send (CTS) and Ground. In a modem only 8 signals and a ground wire are used [53].

Table 2: RS-232 signals [53]

Name	Direction	Type
Signal Common	-	Common
Transmitted Data (TD)	TO DCE	Data
Received Data (RD)	From DCE	Data
Request to Send (RTS)	To DCE	Control
Clear to Send (CTS)	From DCE	Control
DCE Ready (DSR)	From DCE	Control
DTE Ready (DTR)	To DCE	Control
Ring Indicator (RI)	From DCE	Control
Received Line Signal Detector (DCD)	From DCE	Control
Signal Quality Detector	From DCE	Control
Data Signal Rate Detector from DTE	To DCE	Control
Data Signal Rate Detector from DCE	From DCE	Control
Ready for Receiving	To DCE	Control
Remote Loopback	To DCE	Control
Local Loopback	To DCE	Control
Test Mode	From DCE	Control
Transmitter Signal Element Timing from DTE	To DCE	Timing
Transmitter Signal Element Timing from DCE	From DCE	Timing
Receiver Signal Element Timing From DCE	From DCE	Timing
Secondary Transmitted Data	To DCE	Data
Secondary Received Data	From DCE	Data
Secondary Request to Send	To DCE	Control
Secondary Clear to Send	From DCE	Control
Secondary Received Line Signal Detector	From DCE	Control

Modern systems use a line driver to convert from TTL signals to RS-232 signals and back for communicating, they also typically invert the signal to the more commonly used logic 1 represented by a high voltage [53]. The higher voltages gives longer ranges than TTL levels, since induced voltages are smaller by comparison to the signal. The large overhead also makes it less affected by voltage drop in the wires.

2.5.8 RS-422/485

RS-422 is an electrical specification intended to overcome some of the limitations of RS-232 such as low speed and poor noise handling [55]. This is achieved by using differential signaling for the communication. Later RS-485 was introduced to further improve performance and support multiple drivers on a bus [55]. RS-422 can reach speeds up to 10 Mbits/s [56]. In RS-422/485 receivers a voltage difference between line A and B of 200 mV is a logic high and under -200 mV is a logic low [55][56].

RS-422 has a recommended limit of 1.2 km cable length [55], but at a lower data rate. A RS-422 driver supports up to 10 receivers of unit load (minimum 4 kΩ) [55]. If multiple drivers are needed in a system RS-485 is more suited for it [57] with better tolerances for ground potential differences, and can drive up to 32 unit loads. RS-485 is fully backwards compatible with RS-422 [57]. RS-485 uses a unit load of minimum 12 kΩ.

2.5.9 Comparison between communication methods

For the sailing robot a main communication method was needed. This was to be used to communicate between the larger systems of the boat. This method needed to support adding or changing nodes without much extra work, since the hardware configuration of the boat is constantly changing.

RS-232 is only meant for direct communication between two devices, but it supports an extra channel [53] through the Secondary Transmitted Data and Secondary Received Data signals. RS-422 supports up to 10 unit load receivers, but is not suited for multiple transmitters on a single bus [55]. RS-485 supports up to 32 unit load receivers or transmitters [57].

SPI uses direct communication as well, but uses one extra wire to select which device it is talking to. This means SPI can have as many nodes as slave select pins are available on the master device.

I^2C, CAN and USB are all meant as busses with many devices on them. I^2C can theoretically have up to 1016 (10 bit - 8 reserved) devices [58] if they all have unique addresses, but in practice it is limited by the capacitance and resistance of the bus and the drive capabilities of the devices. A USB network is typically limited by the number of USB ports on the device, but can have up to 127 ports via USB hubs [48]. According to Garmin's [46] guidelines a NMEA 2000 network should have no more than 50 devices connected to it. Since not all of the devices will be on the same bus, but split up between them, the device limit will not really a problem in the sailing robot. The Raspberry Pi 3 has only 4 USB ports, so depending on what devices will use USB, a hub might need to be installed, but otherwise no communication bus is likely to have any problems with the number of devices on it.

The farthest distance between nodes in the sailing robot will not be that long, both the mast and hull is about 4 meters long. Since the controlling hardware will be placed centrally, the longest possible distance from it will be the length of the mast, or about 4 meters.

SPI and I^2C are meant for short distance communications between IC chips. Using a cable with low capacitance I^2C can support longer distances before the 400

pF maximum capacitance limit set by the standard is exceeded. RS-232/422/485 are all meant for longer distances communication in the 1000 meter range, and will not be a problem on the sailing robot. The USB standard [48] sets the maximum cable length to 5 meters, this can be extended using USB hubs. In a NMEA 2000 network the distance between any two points on the network should not exceed 100 meters, and any single drop cable should be at most 6 meters [46].

CAN, USB and RS-422 are using differential signals so they are quite resistance to noise interference, while SPI and I^2C are not as good at handling noise induced into the wires.

After some consideration CAN was chosen as the main communication protocol for the sailing robot for its robustness, and availability of off-the-shelf sensors using the CAN/NMEA 2000 interface. CAN also handles most of the error checking for messages in the CAN controller chip, while RS-422/485 is only a electrical standard and does not have any inbuilt error handling for data sent.

2.6 Sensors and collision avoidance

For autonomous sailing navigation some sensors are needed to know the current position of the boat and the state of the weather. A GPS can be used to know where the boat is. A wind sensor is needed to know the wind direction and wind speed for setting the sails, and adjusting the course if the boat needs to go upwind. To know the current heading a compass of some sort is needed. GPS data could be used to determine the current heading, but that only works if the boat is moving fast enough for the GPS to detect movement which might not be the case [5], and GPS position data can be inaccurate [59].

To ensure the boat does not collide with other boats it needs a way to detect them. Since 2004 the International Maritime Organization requires an AIS system on all passenger ships, cargo ships of 500 gross tonnage and upwards, and ships over 300 gross tonnage on international voyages [60]. This requirement makes it possible to avoid most boats using Automatic Identification System (AIS), and let other boat know from a distance that this boat is not manned through its AIS information. An AIS is a transmitter mounted to the boat and transmits the boat's type, size, position, heading, speed and other info about the boat to other boats, coastal stations and planes. The communication between boats is through VHF transmissions, range normally 20-30 nautical miles [61]. The AIS units usually come equipped with their own GPS for determining their position.

Some other kind of detection for boats that does not have an AIS installed will still be needed, as well as for objects such as rocks and people in the water. possible options are camera, radar, lidar, sonar or other such devices.

3 Implementation

This section describes how different parts of the robot have been implemented, and some reasoning behind the decisions.

3.1 Hull info

The boat hull is a 2.4mR class sailboat. It is a small sailing boat that is about 4 meters long, made for one-man sailing. Some of the floating material in the front and rear of the boat has been removed to make room for cables and mechanics. Any extra holes for ropes when sailing with a normal sail have been sealed with silicone. A removable lid has also been made for the opening in the boat, where the sailor normally sits, to try to keep rainwater out. There is a bilge pump mounted in the bottom of the boat for any water that does come in.

A frame made out of aluminium has been fitted inside the boat to have something to fasten hardware boxes, battery and actuators to. The boxes are fastened with straps, and all cables have connectors to facilitate easier removal.

On the rear the solar tracker unit and compass will be mounted. The thermal camera is mounted on the front of the boat.

3.2 Wingsail

The first prototype used a regular sheet sail, but the current model is fitted with a free rotating wingsail. The sheet sail has problems with the sheet moving, aeroelastic collapse or luffing when the sail is at a small angle to the wind making simulations and estimations difficult. The wingsail should be much easier to control and predict. [22][5]

In a study [22] done by Enqvist Teo Johannes, the pros and cons of different sails were weighted against each other and finally the current design was chosen for the sailing robot. Any setup with more than one wing was eliminated because of the extra weight for such a small boat would mean the gained efficiency would be very low, and the center of gravity of the sailing robot would become very high. Similarly, any wingsail requiring external power such as Flettner rotors would be too large and heavy for the sailing robot. Also for wind turbine propulsion to give more force than a wingsail for its size, the wind speed needs to be at least twice the speed of the boat [32].

A sheet sail also has more parts to control. The angle of the sail needs to be controlled quickly with relation to the direction of the wind, and to be able to hold the angle, powerful mechanisms are needed. In high wind speeds the area of the sail should be reduced, because otherwise the boat might capsize.

With a free-rotating wingsail the angle of the servo wing acts like a throttle [23], making the sailing much easier than with a regular sail for which the angle of the sail needs to be set based on the wind. Rapid changes in wind direction also do not affect the boat as much with a free rotating wingsail since it easily changes direction with the wind by itself, instead of having to detect the wind direction and then correcting

the angle of the sail. Drag created by the wingsail in line with the wind is less than a bare mast [5], making it easier to keep the boat in place even in windy conditions.

The wingsail for the sailing robot was built by Svenska Flygfabriken in Sweden and it is made of fiberglass with carbon fiber ribs. It had two servo wings, one in front of the sail and one behind the sail. This is to make the wingsail require less ballast to balance the sail. At this point the front wing has been removed, and the rear wing has been moved further back.

The wingsail is symmetric, making it have the same lift for positive and negative angles of attack, making tacking easier. In his thesis [5] Gabriel Hugh Elkaim references other designs that used asymmetrical wingsails with higher lift than a symmetric wingsail and flipping it end over end when tacking, but this makes the mast more complex and much heavier.

Another benefit of the wingsail is that it is also not as tall as the previous sail, making the sailing robot easier to lift into and out of the water using a crane.

Åland sailing robots also have smaller sailboats that they use for some competitions. They still use regular sheet sails, so most of the hardware and programming is made modular enough to function with both sail types or has different versions for the other configuration.

3.3 Wind vane self-steering

Research has been conducted by Nico Lehtilä and Fredrik Lamberg [62], and they built a wind vane self-steering mechanism for the sailing robot in 2015. It uses a wind vane as a reference to automatically steer the boat. This is not a new invention and have been used in sailing boats for some time.

The boat has an airfoil that can be rotated at any angle around its vertical axis. This airfoil is connected via linkages and gears to a small rudder in the water so that the small rudder will rotate if the airfoil tilts, (see Figure 16). The small rudder can swing freely side to side and is connected to the main rudder of the boat via ropes and pulleys, so that when it moves the main rudder will turn.

When using self-steering the airfoil is set in line with the wind direction. If the boat changes direction relative to the wind, the airfoil will be hit by the wind, tilting it, and turning the small rudder. This causes the boat's movement through water to move the small rudder left or right, pulling the ropes, turning the main rudder. The boat will turn until the airfoil is in line with the wind direction again. With no outer force acting upon the airfoil a counterweight will return it to upright position making the small rudder be centered and therefore the main rudder as well.

The wind vane has an electric motor for setting the direction of the airfoil. It is equipped with a worm gear to keep it in the position without consuming additional power. It is also equipped with a CAN connected position encoder in the housing for keeping track of the position.

The wind vane can be enabled or disabled with an actuator that tightens or loosens the ropes from the small rudder to the main rudder.

Care has to be taken when the wind vane is engaged not to try to power the rudder with the Saildrive, since it is quite powerful and might break something in

Figure 16: The linkage from the airfoil holder to the bottom gear (missing in picture) connected to the servo rudder. The upper part can be rotated. The image is from [62].

the wind vane. Since engaging or disengaging the wind vane is quite slow, a way to measure the position of the actuator must be devised. The first idea was a simple voltage monitor connected to the feedback of the actuator that directly controlled the clutch of the Saildrive, but the actuator turned out to use a hall effect sensor for feedback, so a better solution is probably to connect the feedback to the Arduino and have it count the pulses and turn on or off the clutch at the right moment.

3.4 Propeller

The sailing robot is planned to be equipped with an electric motor with a propeller. This would enable it to move in emergencies if there was no wind. There are still tests to be done to see how large motor it would need and how much power it would consume.

3.5 Hardware and electronics

3.5.1 Navigation unit

The navigation unit handles course planning and sending the control commands to the actuator control unit. Its main part is a Raspberry Pi 3 that runs the main program of the sailing robot. It is connected to a CAN bus shield via SPI for communication with the actuator control unit and the various other sensors and

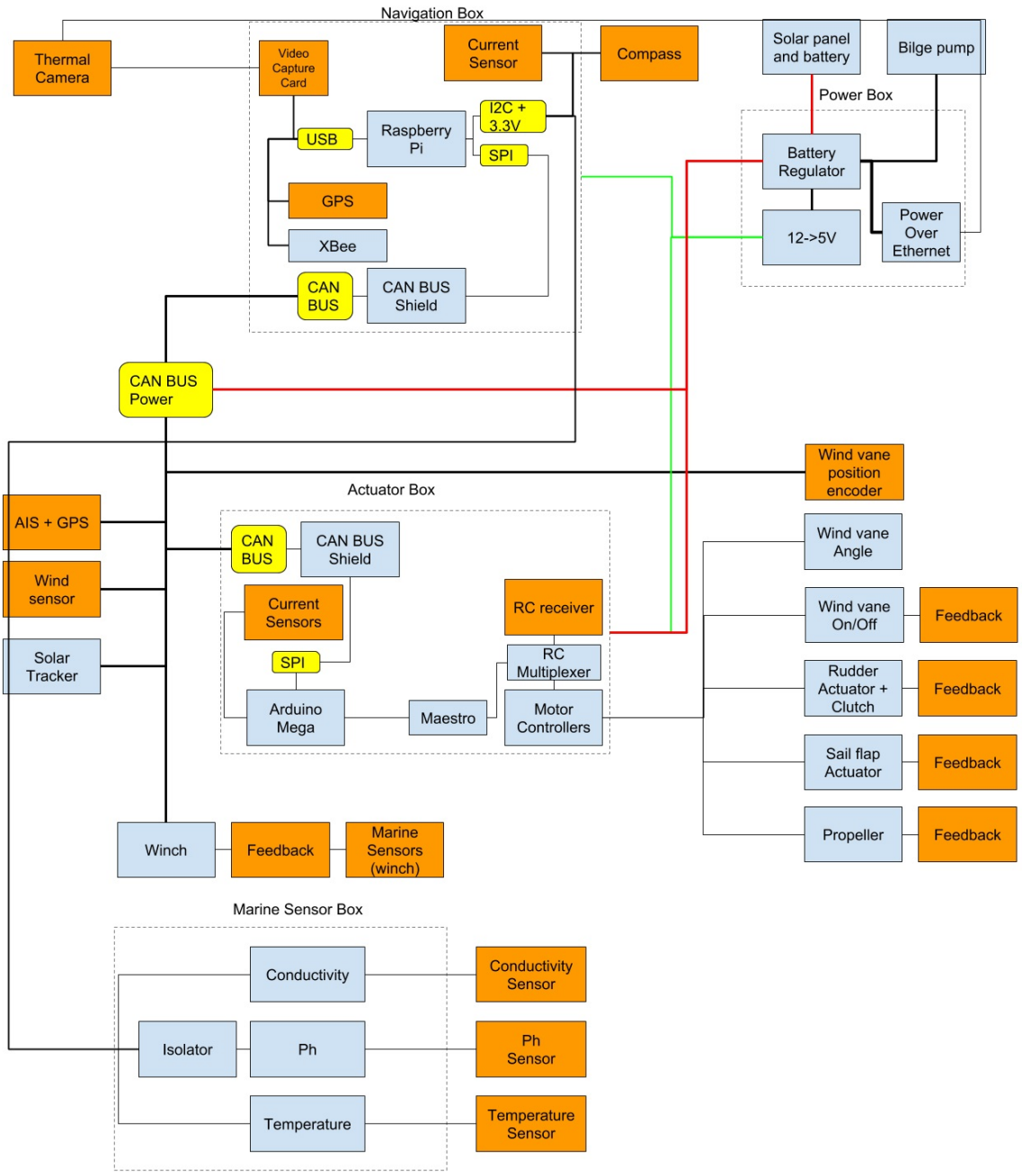


Figure 17: Overview of the hardware in the sailing robot.

motors. It is connected to the marine sensor unit and the compass via I^2C over cat5 network cable. It gets the camera image through a PICAPTURE SD1 video capture card.

The navigation unit will also have a Xbee radio module. This makes it possible to send data it has gathered, and receive data for testing purposes.

Both the navigation unit and the actuator control unit are built into watertight boxes with waterproof connectors for all wires going in and out of them. They also

have current sensors in them for measuring the power consumed by the units.

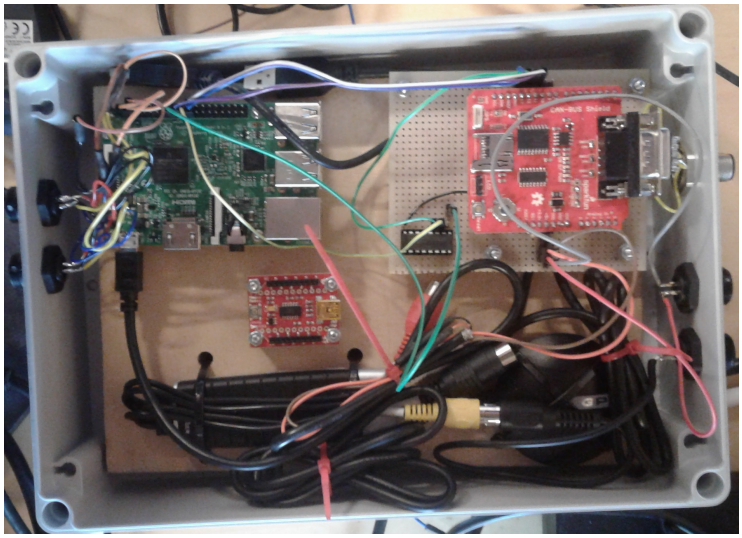


Figure 18: The navigation unit.

3.5.2 Actuator control unit

The actuator control unit directly controls the actuators and receives feedback about their positions. Its main part is an Arduino that receives commands through the CAN bus from the navigation unit.

The Arduino normally controls the motor controllers via a Pololu Maestro Servo Controller that converts the serial commands into pulse-width-modulation signals that gets sent to the motor controllers. It is also equipped with a RC control system for manual control. The signals from the Maestro and RC receiver go into a multiplexer that chooses which input to use as control, normally the Maestro, but one of the RC channels can switch between automatic and manual control. This is set up so that if the radio controller is switched off or loses connection when the sailing robot is in manual mode, it sets the rudder all the way to one side and the sail to neutral position to hopefully bring the boat to a halt faster.

The motor controllers are the Pololu Jrk12V12 motor controllers with 12A continuous current output operating at 12V. They are controlled by a pulse-width-modulation signal. They have their own PID control built in so only the position needs to be set and they handle the monitoring of feedback from the motors. They have a wide variety of options for limits of speeds and positions that can be configured via USB.

3.5.3 Actuators

The linear actuators are used to control the servo wing of the sail and for engaging and disengaging the wind vane. They consist of a regular direct current motor that drives a screw gear that moves the end piece in or out. They also have means to

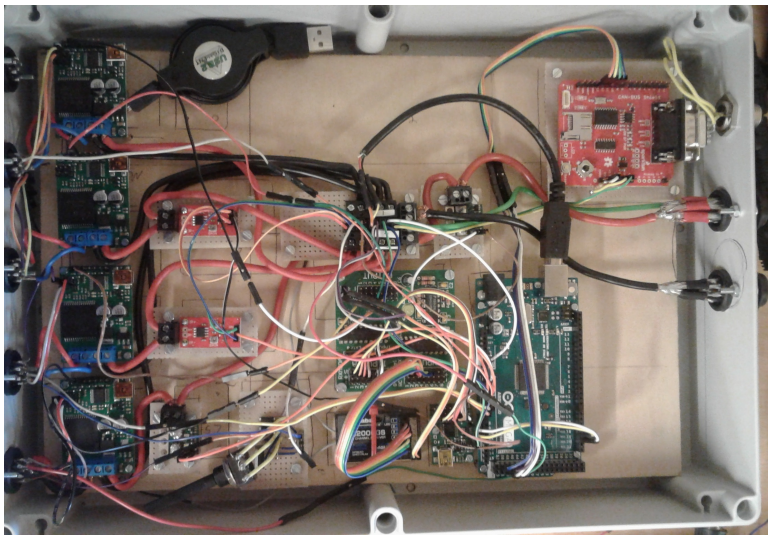


Figure 19: The actuator control unit.

measure the position of the actuator, one uses a potentiometer and the other uses hall effect sensors.

Figure 20: Linear actuator used to control the servo wings, made by Concentric International. The image is from [63].

The servo wings are controlled by lifting an inner shaft in the mast. The shaft is connected to two rods that are each connected to their own levers connected to servo wings. When the shaft moves up and down the servo wings rotate in opposite directions. The actuator is placed vertically under the mast, (see Figure 23).

The wind vane is meant to be enabled and disabled by tightening the ropes connecting the main rudder with the smaller servo rudder. The ropes goes through a pulley system where one end is connected to the linear actuator, and can be moved back and forth.

3.5.4 Saildrive

The OCTOPUS Type RS Saildrive controls the rudder of the sailing robot. It is a commercial autopilot system for boats. It has one wire pair for motor control, and one for engaging the clutch. The clutch is needed to be able to disconnect the rudder from the saildrive and move it using the wind vane instead. It also has an additional module that has a potentiometer for feedback so the position of the rudder can be measured by the Arduino in the actuator control unit. [64]

3.5.5 Marine sensor box

The marine sensor box contains circuits from Atlas scientific for measuring pH, conductivity, and temperature, as recommended as a minimum for measuring marine conditions by one of the researchers that was interviewed. The box also has an isolator that isolates the pH measuring circuit from the other two since the conductivity measuring board might otherwise impact the sensitive pH measurements [65].

The boards communicate with the Raspberry Pi in the navigation unit via I^2C. The I^2C connection is about 1 m long which is much longer than the standard is made for, but we used a cat5 network cable which should give several meters before the 400 pF limit is hit. The signals and voltage supply are all paired with ground lines in accordance with the NXP user manual. A 7 m piece of cable was tested and we confirmed that the sensor circuit boards still worked properly. The I^2C bus in the sailing robot is only run in standard mode (100 kbps) instead of fast mode (400 kbps), this should give more tolerance for higher capacitance since the level transitions have more time.

The sensors are connected to the sensor box with 4 m coaxial cables so that they can be mounted somewhere on the keel of the sailing robot to get the depth necessary for proper measurements.

The data will be stored in a database in the Raspberry Pi and is accessible through a web interface with graphs for easy reading of the measurements while the sailing robot is out gathering data.

3.5.6 Power system

The sailing robot is powered by a 12 volt system. It has a marine 12 volt battery and a solar panel charging it through a solar charge regulator that makes sure the battery is not overcharged. On the connection to the battery there is a cut off switch so that the battery can be disconnected easily when the boat is not in use.

The regulator is placed in the power box. In this box there are also step down converters that convert 12 volts into 5 volts since most of the logic circuits such as the Raspberry Pi and Arduinos in the navigation unit and Actuator Unit run on 5 volts. It was decided to have the conversion happen in one place instead of each box having its own converter. From this box all 12 volt, 5 volt, and ground wires come.

The solar panel is equipped with a Heliomotion [66] solar tracker made by HelioZenit. It uses GPS data, current time, and direction of the boat to calculate the sun's position in the sky relative to the boat.

3.6 CAN bus

The CAN bus was chosen for communication between the navigation unit and the actuation unit since it makes changing and adding new components to the sailing robot easier. The AIS also communicates over CAN bus, forcing us to have a CAN controller in the navigation unit anyway, and the wind vane position encoder sends its position over the CAN bus. An adapter was also acquired for the wind sensor so that it can connect to the CAN bus. The solar panel controller and the winch controller and sensors are connected to the CAN bus.

The AIS and the wind sensor use the NMEA 2000 protocol, while the wind vane position encoder uses CANOpen, a communication protocol using the CAN bus. The communication between the navigation unit and the actuator unit use their own simple message protocols, since they only need to send one or two types of messages. Since the CAN bus will not have many nodes, there will not be that many different messages being sent on the bus. This means one can manually check what messages different nodes send and make sure that their messages will not have the same identification number as any existing messages.

The custom CAN messages for the sailing robot are designed as follows. One message from the navigation unit to the Actuator Unit containing:

- rudder angle,
- wind vane angle,
- wingsail angle,
- speed for the propeller,
- flag indicating if the wind vane should be active or not.

There is also a complementary message from the Actuator Unit to the navigation unit reporting the positions of the actuators, speed of propeller, and if it is in automatic or manual mode. The response message is only used for logging.

There is also a message from the navigation unit to the solar panel control that tells it the current latitude, longitude, time and heading, so it can calculate where the sun is relative to the sailing robot. There is also a message from the navigation unit to the winch controller with the requested depth, and a message from the winch to the navigation unit that reports the current depth. The sensors on the winch will be completely stand alone and store their own collected data, without any other system knowing about them.

3.6.1 Hardware

A Sparkfun CAN-BUS Shield was chosen as the CAN bus interface. It is equipped with a MCP2551 as the CAN transceiver and a MCP2515 as the CAN controller. The MCP2515 implements CAN V2.0B and uses SPI for communication. The MCP2551 handles driving the bus lines and handling faults in the physical wires [68].

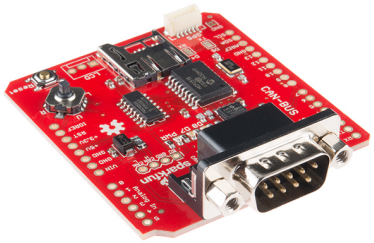


Figure 21: The CAN hardware board used in the sailing robot. The image is from [67].

The controller is equipped with two buffers for receiving messages and three for sending. It handles SPI speeds up to 10 MHz and CAN speeds up to 10 Mbits/s. The controller handles most of the protocol technical details such as error checking, arbitration and waiting to send messages and sending ACK and errors. [69] The only thing the user is left with is the data and the id of the message.

3.6.2 Software

The driver was written in C++ for the Raspberry Pi to handle the communication over the CAN bus using the Sparkfun CAN-BUS Shield. The functions outlined in the MCP2515 CAN controller datasheet were implemented, and the CAN bus was set up for NMEA 2000 communication.

The driver was written in two layers. One low level that implements the MCP2515 functions directly via SPI using the Raspberry Pi IO library WiringPi [70], and a higher level part that uses the low level functions to create an API to be used by other programs. Some helper functions were also written to convert the id of a CAN message into a NMEA 2000 format message.

```
void     MCP2515_SendByte(uint8_t Data);
uint8_t  MCP2515_Read(uint8_t Address);
void     MCP2515_Write(uint8_t Address, uint8_t Data);
uint8_t  MCP2515_ReadStatus();
uint8_t  MCP2515_RXStatus();
void     MCP2515_BitModify(uint8_t Address, uint8_t Mask, uint8_t Data);
bool     MCP2515_Init();
uint8_t  MCP2515_CheckMessage();
uint8_t  MCP2515_CheckFreeBuffer();
bool     MCP2515_GetMessage(CanMsg *message, uint8_t MsgAddress);
uint8_t  MCP2515_SendMessage(CanMsg *message, uint8_t MsgAddress);
void     MCP2515_OutputInfo();
```

Listing 1: Low level functions

Basic functions

The MCP2515_SendByte command just sends a byte of data over the SPI bus, this is just a helper function used by the other commands.

The MCP2515_Read command reads the MCP2515 chips internal register at the specified address, and the MCP2515_Write command writes to the MCP2515 chip's internal register. The MCP2515_ReadStatus command reads the status register of the MCP2515. This gives information if any messages have been received and if outgoing messages have been sent. Similarly the MCP2515_RXStatus command is used to find out information about the current received messages, such as what filter matched them and what type of messages they are (standard, extended, remote). The MCP2515_BitModify command is used to modify specific bits in registers. The supplied Mask field chooses which bits to modify and the Data field what values they should have.

The commands use the wiringPiSPIDataRW(Channel, ReadWrite, Bytes) function from the WiringPi library for writing over the SPI bus. The Bytes field controls how much data gets clocked out on the SPI bus on the specified channel, it reads from the ReadWrite buffer and then writes the incoming data to the same buffer.

The MCP2515_Init function resets the MCP2515, putting it to configuration mode, and setups its speed for a NMEA 2000 messages, 250 kbits/s. After writing the initial setup registers it tries to read one of them back to see if the chip is responding. Then it disables all filters and activates message rollover from buffer 0 to 1 if full. Lastly it sets the MCP2515 to normal mode.

The MCP2515_OutputInfo function writes the values of the control and interrupt registers, the status of the error flags, and the output pin status. This is used for debugging.

Sending and receiving messages

The MCP2515_CheckMessage function checks if there are any messages waiting in the buffers of the MCP2515. It uses the RXStatus command to check how many messages are waiting, if there are none, it returns 0. If there is one it returns the address of the buffer containing the message. If there are messages in both buffers the function will alternate and return the address one of them, thereby guaranteeing no message will be blocked. The MCP2515_GetMessage function will retrieve the message from the buffer with the address given, or if no address is given it will call CheckMessage to see if there are any and retrieve one if there are, and place it at the address pointed to by the message pointer. The message is in a CanMsg struct that contains the id of the message, a field indicating if the message is a extended or standard frame, the length of the data section and the data itself. The function returns true if it found a message and false if there was none.

The MCP2515_CheckFreeBuffer function calls the ReadStatus command and checks if there are any free buffers for sending messages, and if there are it returns the address of a free buffer. If there are no free buffers it returns 0. The MCP2515_SendMessage function sends the message using the buffer with the supplied address, or calls CheckFreeBuffer of no address is given. It writes the message to the transmit buffer of the MCP2515 and sends a Request To Send message to the chip. It returns the address of the buffer used if successful and 0 if it failed.

```
//In both classes
bool    Init(int SPISpeed);
```

```
uint8_t CheckForMessages();
bool    SendMessage(CanMsg *Msg);
bool    GetMessage(CanMsg *Msg);
//NMEA2000Bus
void    IdToN2kMsg(N2kMsg &NMsg, uint32_t &id);
void    N2kMsgToId(N2kMsg &NMsg, uint32_t &id);
int GetN2kMsg();
bool    IsFastPackage(const N2kMsg &Msg);
bool    ParseFastPKG(CanMsg &Msg, N2kMsg &NMsg);
//CanbusClass
void    SetNormalMode();
void    SetSleepMode();
void    SetLoopBackMode();
void    SetListenOnlyMode();
void    SetConfigMode();
void    SetMasksAndActivateFilters(uint32_t Mask1, uint32_t Mask2, bool RollOver);
void    SetFilter(int FilterIndex, uint32_t Filter);
void    SetFilterAndMask(int ReceiveBuffer, int FilterIndex, uint32_t Filter, uint32_t Mask);
```

Listing 2: High level functions

I created a CanbusClass and a NMEA2000Bus class to handle the communication over the CAN bus. The NMEA2000Bus class translates the NMEA 2000 messages and contains a CanbusClass object that it uses when communicating.

The Init function calls wiringPiSetup to start the WiringPi library, then it setups SPI bus with the correct channel and speed and checks that it succeeds. Then it calls the MCP2515_Init function.

The SetNormalMode, SetSleepMode, SetLoopBackMode, SetListenOnlyMode and SetConfigMode set the MCP2515 into the corresponding mode. SetFilterAndMask allows changing the filters of the MCP2515 to filter out message ids, this is currently not used in the sailing robot.

Sending and receiving NMEA 2000 messages

CheckForMessages, SendMessage and GetMessage calls the MCP2515 versions. The CheckForMessages could also check the value of a physical pin on the MCP2515, which would be slightly faster. Many of the functions simply calls the MCP2515 versions of the functions, but I wanted to make it so that the hardware implementation could be changed without having to have to rewrite code higher up. If the NMEA 2000 version of the SendMessage is called it takes a N2kMsg instead of a CanMsg. The function then converts it into a CanMsg and calls the CanbusClass SendMessage with that. It does not support sending fast packages.

The N2kMsg struct contains the PGN number, the priority of the message, the Source address, destination address, length of data in bytes and the data. While the CanMsg struct can have a maximum of 8 bytes of data, the N2kMsg struct has no limit, except for the 223 bytes limit in the NMEA 2000 standard.

IdToN2kMsg takes a CAN message id and converts it into a N2kMsg struct. There is also the inverse function N2kMsgToId that takes a N2kMsg struct and returns an id value.

IsFastPackage is a simple switch case function that compares the PGN of a N2kMsg with a set list of known FastPackages in use by the hardware in the sailing robot to determine if the received message is a fast package or not.

ParseFastPKG extracts the sequence id and number of the fast package. It combines the message id and the sequence id into a key that it uses to see if it has gotten any previous parts of this fast package. If this is the first part it creates a

new N2kMsg and set the data length and allocates memory for the data to the value from the first part of the fast package. Then it copies the 6 bytes from the first package. If the data length is 6 or less, it has the full package and returns it. If not then it saves the partial message and how many bytes are left of the message, in a map using the key that it made.

If this is not the first part of a message the function receives, it checks how many bytes are left of the message and if it is more than 7 it copies all data from the package and decreases the bytes left counter for the message. If it is less than 7 bytes left it only copies the needed bytes, removes the key and returns the completed message.

The GetN2kMsg is the NMEA 2000 version of GetMessage, but it is more complicated than the normal one. It first calls the normal GetMessage and converts the id into a PGN using IdToN2kMsg. Then it checks if it is a fast package using the function IsFastPackage. If it is not it simply copies over the data length and data from the regular CAN message and puts the NMEA 2000 message in a queue and returns the value 1.

If the message is a fast package, the function calls ParseFastPKG. If the message has been received in full it is put into a queue and the value 1 is returned, if only a partial message has been received the value 0 is returned.

3.7 Sensors

The boat is equipped with a suite of sensors both for navigation and data gathering.

3.7.1 Winch sensors

The sailing robot is planned to have a winch with sensors on the end such as a hydrophone on it to locate harbor porpoises. But at the time of writing this is currently on hold. The hydrophone will detect the echolocation clicks the harbour porpoises send out to hunt food.

3.7.2 Navigation

As wind sensor an LCJ Capteur CV7 has been bought for the sailing robot, and a WindyPlug adapter so it can be connected to the CAN bus. The wind sensor measures wind speed and wind direction, while the plug has sensors for temperature and air pressure. It get its power from the CAN bus. The wind sensor uses no moving parts, instead it has four ultrasonic transducers that it uses to calculate the wind speed and direction. This makes it more robust and reliable, since moving parts are more prone to failure [71][72]. The wind sensor is mounted at the top of the mast.

The sailing robot will be equipped with a FLIR MD-324 Thermal Imager camera mounted in the front. The plan is to detect boats from the heat of the engines, and humans. It will be used for detection of all objects close to the sailing robot that do not show up on AIS. Other detection methods have been looked into but found to be unsuitable for this sailing robot. Radar data would be skewed by fast turning of the vessel and sonar would be too expensive. Both would also consume a lot of power.

The AIS CTRX GRAPHENE was chosen as the AIS unit for the sailing robot. It is a class B AIS transponder. It comes equipped with its own GPS. [61]

As a compass the Honeywell HMC6343 3 axis compass has been chosen. It is mounted in a separate box away from other electronics that might affect it, and it communicates with the navigation unit using I^2C over cat5 network cable. In addition to heading it can measure tilt and roll as well. This makes it more accurate since it can compensate for not being perfectly tangential to the earth's surface. [73]

3.8 Collision avoidance and navigation logic

Currently the sailing robot is equipped with GPS for knowing its position, AIS for avoiding other boats, and a thermal camera for detecting things close to the sailing robot.

The AIS gives detailed information of the other boats position, course, speed, rate of turning, and size among other things [74], while objects picked up by the thermal camera will only reliably have the bearing of the object compared to the direction of the boat. Distance and speed would have to be estimated using complicated techniques where most of them needs to move the camera to capture the object either at different distances and or angles, or know the size of the object the camera is trying to measure the distance to [75]. Distance measurements with a single fixed camera can also be difficult on a boat that will roll, tilt, and pitch with the waves.

The sailing robot does not assume that other boats will avoid it, so the strategy is to keep sufficient distance to all other boats. Avoiding collisions with other boats or objects in the water has a higher priority than navigating towards the next waypoint.

Voter system

The sailing robot uses a voter system to decide what course to set. It currently has a waypoint, wind, channel, proximity, and a mid-range voter. Each voter gets a number of votes to spend on what courses it thinks are the best out of the possible 360, then the system uses the most voted for course.

The simplest of them is the waypoint voter, that simply favours courses that are toward the waypoint. The wind voter keeps the sailing robot from turning straight into the wind and votes instead for tacking angles. It also prioritizes courses similar to the current course to avoid turning too often and losing speed, which was a problem early in the simulations when it tended to do very abrupt turns to avoid things and ending up much closer than if it kept a straight course.

The channel voter tries to keep the sailing robot within a distance of the vector between waypoints to avoid getting into shallow waters. It also causes the sailing robot to switch tacking sides when the boat gets close to the edges, since the wind voter gives equal votes to both tacking angles but gives extra weight to keeping the current course.

The proximity voter is supposed to take evasive maneuvers when the robot gets too close to another boat or other object. It chooses a course perpendicular to the

other boat's course. It is idle if nothing is within its safe range, currently set at 150 meters.

The mid-range voter calculates where it would be relative to other boats based on their courses, for a given course. It gives all courses equal votes to start and then removes votes based on how close to the other boat it would get.

The different voters also have different weight for how the votes are counted, for example, the proximity voters votes counts double and therefore almost always wins if another boat gets too close. [76]

3.9 Software architecture

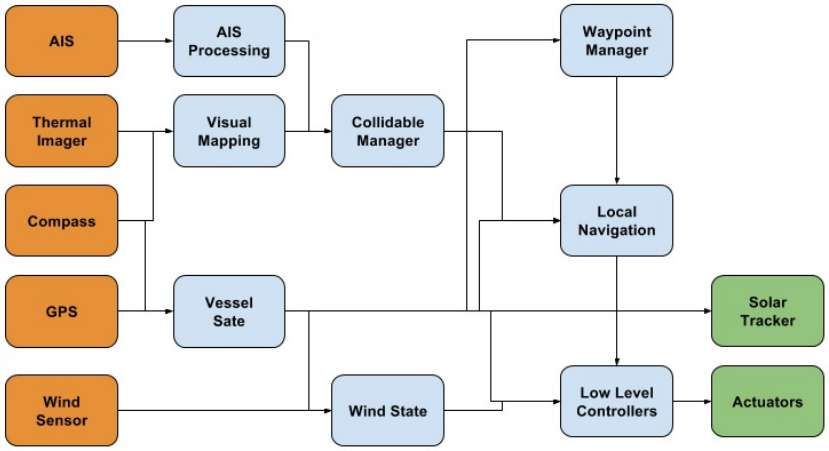


Figure 22: Simplified overview of the software architecture. The orange nodes are sensors, the blue nodes are part of the software, and the green nodes are other hardware nodes.

The code running in the sailing robots hardware is mainly written in C/C++. It was chosen since it is a compiled language and therefore most bugs can be discovered at compile-time instead of later when running the code and the sailing robot is possibly out on the water [76].

Internally the system has a message bus that handles all communication between different parts of the program. It has passive and active nodes, passive nodes only listen to the bus and active nodes run in a separate thread and produce messages on the bus.

This makes the system more modular and easier to change since no nodes should know about other nodes, only message and node IDs. The nodes can subscribe to receive specific messages or message types on the bus.

A message consists of a message ID to uniquely identify the message, message type for casting and subscription, source ID where the message came from, and destination ID for sending to a specific node, or to everyone.

4 Results

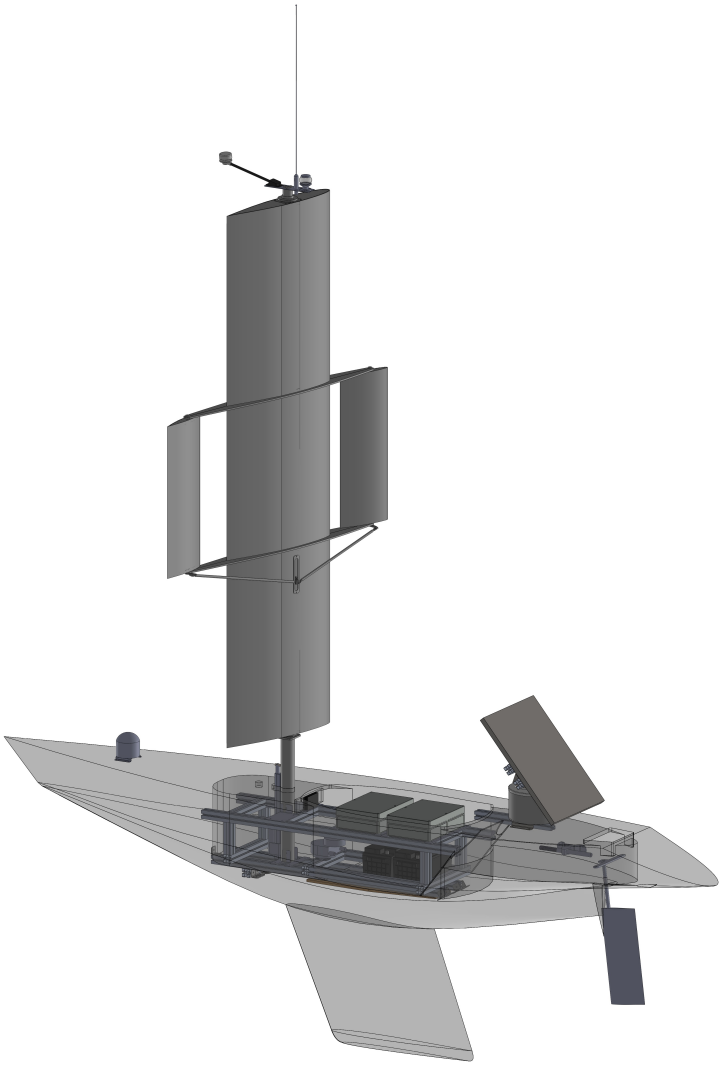


Figure 23: Overview of the boat where the wing mechanism can be seen.

4.1 Navigation simulations

Simulations of the voters can be seen in Figure 24 and 25. In Figure 24 the sailing robot is tacking against the wind, and where the proximity voter takes over and goes outside the channel can clearly be seen. In Figure 25 it is avoiding 3 other boats and has to go outside the channel to do so, and still getting within 30 m of one of the boats. Note that the background is not part of the simulations.

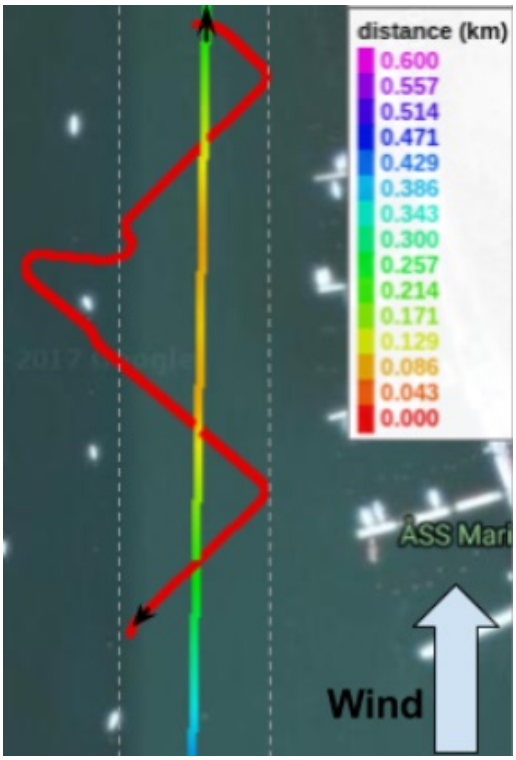


Figure 24: Simulation of the voting system with one other boat. The red line is the course the sailing robot took, the other lines are the other boats and their color represents the distance to the sailing robot at that point. The dotted lines shows the channel voters edges. The sailing robot starts at the top and reaches its waypoint at the end of the red line. Wind direction is south to north. The image is from [76].

4.2 First test of wingsail (7.6.2017)

A test was performed with the wingsail on the boat in water. Only the actuator box was mounted in the boat and it was operated via remote control. The only electronics connected was the sail angle actuator and the rudder actuator. The test was successful and the boat was able to be steered, but a plug in the keel of the boat was missing so the boat quickly filled up with water and the test had to be aborted so that the boat could be brought onto land to drain.

The windsail worked but seemed to not be correctly balanced, since when the servo wings were kept straight the sail seemed to have two balance points, with the wind not changing. One correct and one almost perpendicular to the wind. The boat also did not tack well up-wind, and started to go backwards most of the time.

After some more testing with a smoke machine it was concluded that the front servo wing was incorrectly designed, and a decision to remove it and put in a counter weight in its place was made in hope of having a better functioning wingsail. The aerodynamic centerline was slightly in front of the mast making the wingsail behave erratically.

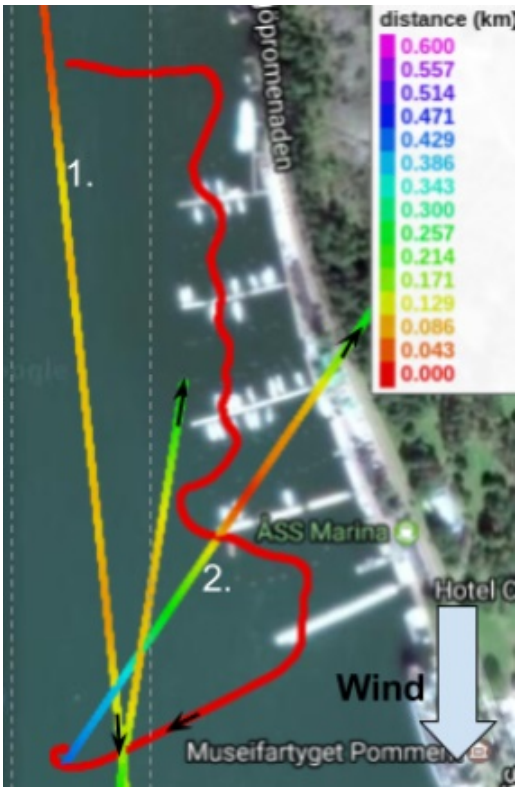


Figure 25: Simulation of the voting system with three other boats with differing speeds. The sailing robot starts at the top and reaches its waypoint at the end of the red line. Wind direction is north to south. The image is from [76].

4.3 WRSC 2017

The sailing robot won WRSC 2017 that took place in Horten, Norway in September. Here are some results from the different tests. At that time the wing sail had been modified, moving the rear servo wing further back, and replacing the front servo wing with a counterweight. The self-steering mechanism was not mounted, and the collision avoidance logic was not used.

The boats were equipped with data loggers that recorded their GPS coordinates at least once a second. The images are the recorded data displayed using the tool on their website.

In the race the boats were supposed to go from the starting line around two buoys and back. In Figure 26 attempt one and two in different wind conditions can be seen. Here the tacking maneuvers made by the sailing robot can clearly be seen.

In the area scanning test the boats were supposed to cover an area as fast as possible. The area was divided into 20 x 20 meter squares, and the boat is supposed to visit as many squares as possible in a certain time limit. The area to cover was L shaped, and can be quite clearly seen by the path the boat took in Figure 27.

In the station keeping contest the boats have to try to stay as close as possible to a coordinate point for 5 minutes. After entering within 20 meters of the point the

timer is started. The smallest possible circle containing 95% of the points logged during the 5 minutes is the score. Results from this challenge can be seen in Figure 28.

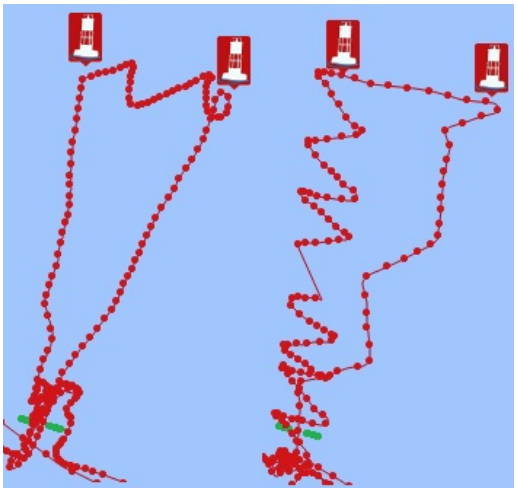


Figure 26: WRSC race attempt number one and two made by the sailing robot. The attempts had different wind conditions and times when the boat was tacking can clearly be seen. The image is from [77].

Figure 27: WRSC area scanning, the area to cover was L-shaped. The image is from [77].

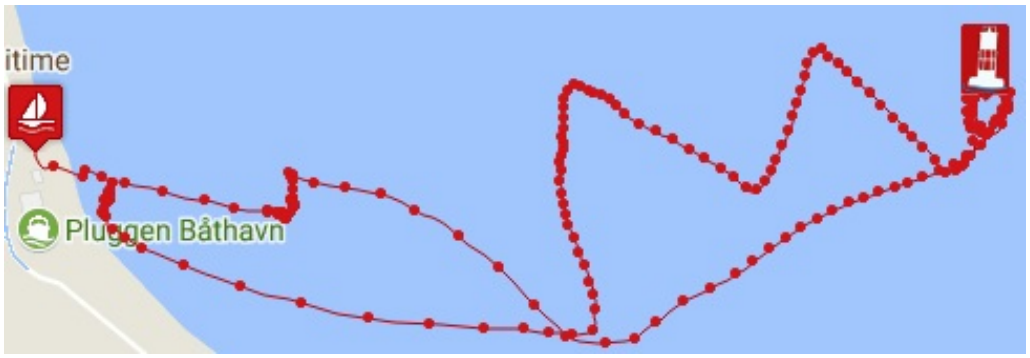


Figure 28: WRSC station keeping, with traveling to and from the point included. The image is from [77].

5 Discussion

This thesis aims to provide an overview of the state of the sailing robot at the time of writing. The development of the sailing robot is continually moving forward and by the time you are reading this some information written here might not be accurate for the project anymore. Overall I hope this thesis gives insight into some of the problems and solutions in the world of robotic sailing.

I presented some other similar projects, and then some alternatives for propulsion, communication and sensors. The sailing robot did initially have an regular sheet sail, but later changed to a wingsail.

I think the wingsail will be successful but it was a mistake going for a more complex wingsail with both front and rear servo wings, over the standard with only a rear servo wing. The wingsail with two servo wings turned out to not be properly balanced, the aerodynamic center point was slightly in front of the mast instead of behind it. I think it would have been easier to balance it with only the rear servo wing. Alternatively it needs a new smaller servo wing mounted in the front or a larger servo wing in the back, if it turned out that the deflection angle is too small with only the current rear servo wing in place.

The front servo wing was later removed and the rear wing was moved further back to achieve a working system.

The single camera is yet to be tested if it will be an effective way to avoid obstacles. If it is replaced with two cameras, distance measurements would be much easier to do, but using two thermal cameras would be expensive so using two different cameras might be tested. The necessity for thermal cameras is also not clear yet, and it is still uncertain if distance is necessary for avoiding objects in the real world scenarios. The sailing robot overall has had very little real world collision avoidance testing. That would give valuable data in how the boat handles in the water.

The sailing robot seems overall to be quite successful, with the results in the WRC 2017 showing that it can navigate quite well both with and against the wind, and at least in the simulations it can avoid collision most of the time.

References

[1] The microtransat challenge. http://www.microtransat.org/. [Online; accessed September-2017].

[2] Åland University Of Applied Sciences. Åland sailing robots. http://sailingrobots.ax/. [Online; accessed June-2017].

[3] World robotic sailing championship & international robotic sailing conference. http://www.roboticsailing.org/. [Online; accessed September-2017].

[4] WRSC & IRSC 2017. https://www.wrsc2017.com. [Online; accessed September-2017].

[5] Gabriel Hugh Elkaim. *System identification for precision control of a wingsailed GPS-guided catamaran*. PhD thesis, Stanford University, 2001.

[6] University of New Hampshire. Charter and rate information. https://marine.unh.edu/rv-gulf-challenger-charter-and-rate-information. [Online; accessed August-2017].

[7] Monterey Bay Aquarium Research Institute. Rates for vessels, vehicles, mars, labor, test tank. http://www.mbari.org/at-sea/mars-ship-rates/. [Online; accessed August-2017].

[8] University of connecticut. Vessel rates. http://marinesciences.uconn.edu/mstc/vesselops/rates/. [Online; accessed August-2017].

[9] Marine Institute. Vessel rates. https://www.marine.ie/Home/site-area/infrastructure-facilities/research-vessels/vessel-rates. [Online; accessed August-2017].

[10] Saildrone. http://saildrone.com/. [Online; accessed June-2017].

[11] Saildrone. http://saildrone.com/missions/gulf-of-mexico-2. [Online; accessed February-2018].

[12] C-enduro. https://www.asvglobal.com/product/c-enduro/. [Online; accessed June-2017].

[13] AutoNaut. http://www.autonautusv.com/. [Online; accessed June-2017].

[14] Eirik Bøckmann. *Wave propulsion of ships*. PhD thesis, Norwegian University of Science and Technology, 2015.

[15] Liquid Robotics. Wave glider. https://www.liquid-robotics.com/platform/overview/. [Online; accessed June-2017].

[16] Liquid Robotics. Wave glider. https://www.liquid-robotics.com/platform/how-it-works/. [Online; accessed June-2017].

[17] Ocean Aero. Submaran™s10: Wind and solar-powered freedom to go further and faster. http://www.oceanaero.us/Ocean-Aero-Submaran. [Online; accessed June-2017].

[18] Frank Cassidy. *A companion to the archaeology of the ancient Near East*. Wiley-Blackwell, 2012.

[19] Bryon D. Anderson. The physics of sailing. *Physics Today*, 61(2), February 2008.

[20] Charles N Eastlake. An aerodynamicist's view of lift, Bernoulli, and Newton. *The Physics Teacher*, 40(3):166–173, 2002.

[21] RW Bilger. A simplified theory for the tacking of a sailboat. In *Proc. Twelfth Australasian Fluid Mech. Conf*, 1995.

[22] Enqvist Teo Johannes. Wingsail arrangements for robotic sailboat. Thesis project, Åland University Of Applied Sciences, 2016.

[23] Adam Fisher. The drone that will sail itself around the world. https://www.wired.com/2014/02/saildrone/, 2014. [Online; accessed June-2017].

[24] Lloyd Bergeson and C. Kent Greenwald. Sail assist developments 1979-1985. *Journal of Wind Engineering and Industrial Aerodynamics*, (19):45–114, 1985.

[25] Rdurkacz. Image of Magnus effect. https://commons.wikimedia.org/w/index.php?curid=24490137. CC BY-SA 3.0, [Online; accessed September-2017].

[26] A. De Marco, S. Mancini, C. Pensa, G. Calise, and F. De Luca. Flettner rotor concept for marine applications: A systematic study. *International Journal of Rotating Machinery*, 2016.

[27] Popular Science. New rotor ship sails in lightest wind, July 1933.

[28] J. Borg. Magnus effect: An overview of its past and future practical applications, 1850-1985, volumes 1 and 2. *NASA STI/Recon Technical Report N*, 86, 1986.

[29] S.J. Savonius. *The Wing-rotor in Theory and Practice*. Omnia-Mikrofilm-Technik, 1981.

[30] M Ragheb. Wind energy converters concepts. *University of Illinois at Urbana-Champaign, Champaign, IL*, 2014.

[31] B. L. Blackford. Optimal blade design for windmill boats and vehicles. *Journal of Ship Research*, 29(2):139–149, June 1985.

[32] Eirik Bøckmann and Sverre Steen. Wind turbine propulsion of ships. *Second International Symposium on Marine Propulsors, smp'11*, 2011. Department of Marine Technology, Norwegian University of Science and Technology (NTNU), Trondheim, Norway.

[33] Motorola, Inc. *SPI Block Guide*, 2003. V03.06.

[34] Byte Paradigm. Introduction to i^{2}c and SPI protocols. http://www.byteparadigm.com/applications/introduction-to-i2c-and-spi-protocols/?/article/AA-00255/22/Introduction-to-SPI-and-IC-protocols.html. [Online; accessed August-2017].

[35] Corelis. SPI tutorial. http://www.corelis.com/education/SPI_Tutorial.htm. [Online; accessed August-2017].

[36] NXP Semiconductors. *I^{2}C-bus specification and user manual*, 2014. Rev. 6.

[37] GLOOMYANDY. I2C on the EV3. https://lejosnews.wordpress.com/2016/01/15/i2c-on-the-ev3/. [Online; accessed September-2017].

[38] Steve Corrigan. *Introduction to the Controller Area Network (CAN)*. Texas Instruments, 2002. Revised May 2016.

[39] BOSCH. *CAN Specification Version 2.0*, 1991. Revised March 1997.

[40] MOTOROLA. *Bosch Controller Area Network (CAN) Version 2.0 Protocol Standard*, 1998. Rev. 3.

[41] Markus Junger. *Introduction to J1939*, 2010. Version 1.1.

[42] Yuan Zhu, Hao Wu, Guangyu Tian, Xianhui Yang, Zhao Li'an, and Weibo Zhou. Control and communication network in hybrid fuel cell vehicles. *TsingHua science and technology*, 9(3):345–350, 2004.

[43] National Instruments. J1939 transport protocol reference example. http://www.ni.com/example/31215/en/. [Online; accessed August-2017].

[44] Frank Cassidy. *NMEA 2000 Explained - The Latest Word*. National Marine Electronics Association, 1999.

[45] Lee A. Luft, Larry Anderson, and Frank Cassidy. *NMEA 2000 A Digital Interface for the 21st Century*, 2002. Presented at the Institute of Navigation's 2002 National Technical Meeting January 30, 2002 in San Diego, California.

[46] Garmin. *NMEA 2000 Network Fundamentals*, 2008. Rev. A.

[47] Compaq, Hewlett-Packard, Intel, Lucent, Microsoft, NEC, and Philips. *Universal Serial Bus Specification*, 2000. Rev. 2.0.

[48] Hewlett-Packard Company, Intel Corporation, Microsoft Corporation, Renesas Corporation, ST-Ericsson, and Texas Instruments. *Universal Serial Bus 3.1 Specification*, 2013. Rev. 1.0.

[49] FTDI chip IC product site. http://www.ftdichip.com/Products/ICs.htm. [Online; accessed September-2017].

[50] Future Technology Devices International Ltd. *FT232R USB UART IC*. FT232R Datasheet, Version 2.13.

[51] *USB to serial chip CH340*. CH340 datasheet, Version 1D.

[52] Silicon labs. *Single-chip USB-to-UART bridge*. CP2102 datasheet, Rev. 1.8.

[53] Dallas semiconductor. *Fundamentals of RS–232 Serial Communications*, 1998. Application Note 83.

[54] Texas Instruments. *Interface Circuits for TIA/EIA-232-F*, 2002.

[55] Michael R. Wilson. *TIA/EIA-422-B Overview*. National Semiconductor, 2000. Application Note 1031.

[56] Hein Marais. *RS-485/RS-422 Circuit Implementation Guide*. Analog devices, 2008. Application Note AN-960, Rev. 0.

[57] Texas Instruments. *Comparing EIA-485 and EIA-422-A Line Drivers and Receivers in Multipoint Applications*, 2004. Application Note 759.

[58] Philips Semiconductors. *AN10216-01 I^2C Manual*, 2003.

[59] National Coordination Office for Space-Based Positioning, Navigation, and Timing. GPS accuracy. http://www.gps.gov/systems/gps/performance/accuracy/. [Online; accessed August-2017].

[60] International Maritime Organization. AIS transponders. http://www.imo.org/en/OurWork/Safety/Navigation/Pages/AIS.aspx. [Online; accessed August-2017].

[61] True Heading. *AIS CTRX Graphene class b AIS transponder manual*, 2013. Version 1.4E.

[62] Nico Lehtilä and Fredrik Lamberg. Styranordning för autonoma segelbåtar. Thesis project, Åland University Of Applied Sciences, 2015.

[63] Concentric International. Linear actuator 6 inch stroke (nominal), 110 lb force, 12 vdc. http://www.concentricintl.com/product/linear-actuator-5-90-150-mm-stroke-107-lb-force-12-vdc/. [Online; accessed September-2017].

[64] Octopus. *Type RS Sailboat Drive INSTALLATION MANUAL*, 2009. Rev. A.

[65] Atlas scientific webpage. https://www.atlas-scientific.com/. [Online; accessed September-2017].

[66] Heliozenits heliomotion web page. https://heliomotion.com/. [Online; accessed September-2017].

[67] SparkFun Electronics. Sparkfun CAN-bus shield store page. https://www.sparkfun.com/products/13262. [Online; accessed October-2017].

[68] Microchip. *MCP2551 High-Speed CAN Transceiver*, 2016. MCP2551 datasheet.

[69] Microchip. *MCP2515 Stand-Alone CAN Controller with SPI Interface*, 2016. MCP2515 datasheet.

[70] Wiring pi webpage. http://wiringpi.com/. [Online; accessed September-2017].

[71] Taylor Barton and Mariano Alvira. A discrete-component 2d-array wind sensor without moving parts for a robotic sailboat. In *Robotic Sailing 2012*, pages 95–104. Springer, 2013.

[72] *Ultrasonic wind sensor*. CV7 manual.

[73] *3-Axis Compass with Algorithms*. HMC6343 datasheet.

[74] NMEA. NMEA 2000® parameter group descriptions (messages) with field description. https://www.nmea.org/Assets/20151026%20nmea%202000%20pgn_website_description_list.pdf. Version 2.101, [Online; accessed August-2017].

[75] Peyman Alizadeh. Object distance measurement using a single camera for robotic applications. Master's thesis, Laurentian University, 2015.

[76] Jordan Less'ard-Springett. Local navigation for autonomous sailboats. Bachelor's thesis, Aberystwyth University, 2017.

[77] WRSC & IRSC 2017 replay. http://tracking.wrsc2017.com/replay. [Online; accessed September-2017].